AD-A237 192

NAVAL POSTGRADUATE SCHOOL
Monterey, California

DTIC
ELECTE
JUN 21 1991
S B D

THESIS

CONTRACTING FOR EMBEDDED COMPUTER SOFTWARE WITHIN THE DEPARTMENT OF THE NAVY

by

Henry Attanasio

June 1990

Thesis Advisor: Martin J. McCaffrey

Approved for public release; distribution unlimited

91-02441

91 6 18 030

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

1a REPORT SECURITY CLASSIFICATION: Unclassified	1b RESTRICTIVE MARKINGS
2a SECURITY CLASSIFICATION AUTHORITY	3 DISTRIBUTION/AVAILABILITY OF REPORT: Approved for public release; distribution is unlimited.
2b DECLASSIFICATION/DOWNGRADING SCHEDULE	
4 PERFORMING ORGANIZATION REPORT NUMBER(S)	5 MONITORING ORGANIZATION REPORT NUMBER(S)
6a NAME OF PERFORMING ORGANIZATION: Naval Postgraduate School; 6b OFFICE SYMBOL (If Applicable): Code 36	7a NAME OF MONITORING ORGANIZATION: Naval Postgraduate School
6c ADDRESS (City, State, and ZIP Code): Monterey, CA 93943-5000	7b ADDRESS (City, State, and ZIP Code): Monterey, CA 93943-5000
8a NAME OF FUNDING/SPONSORING ORGANIZATION; 8b OFFICE SYMBOL (If Applicable)	9 PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER
8c ADDRESS (City, State, and ZIP Code)	10 SOURCE OF FUNDING NUMBERS: PROGRAM ELEMENT NO. / PROJECT NO. / TASK NO. / WORK UNIT ACCESSION NO.

11 TITLE (Include Security Classification): CONTRACTING FOR EMBEDDED COMPUTER SOFTWARE WITHIN THE THE DEPARTMENT OF THE NAVY

12 PERSONAL AUTHOR(S): ATTANASIO, HENRY

13a TYPE OF REPORT	13b TIME COVERED	14 DATE OF REPORT (Year, Month, Day)	15 PAGE COUNT
Master's Thesis	From To	1990 June	101

16 SUPPLEMENTARY NOTATION: The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

17 COSATI CODES			18 SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB-GROUP	Software development; Software acquisition; Contracting for Software development; Mission Critical Computer Resources (MCCR); Embedded computer Resources (ECR)

19 ABSTRACT (Continue on reverse if necessary and identify by block number)

Over the past 15 years the reliance on computer software, especially within critical weapons systems, has grown in orders of magnitude. Government reports during this same period, however, consistently cite the poor success rate experienced by the Department of Defense in contracting for the development of specialized software as an area of great concern. This thesis investigates the guidance provided by the Department of the Navy, and the contractual methods available, to program managers and contracting officers regarding the procurement of custom designed software. The research conducted concludes that the majority of the blame for software procurement problems should fall upon the personnel within the acquisition system. The problem is a problem of "attitude" concerning software development. Although the system is admittedly cumbersome, the underlying causes resulting in cost overruns, schedule delays, and poor performance are not with the mechanics of the procurement system. Program managers and contracting officers currently have the tools available within the acquisition system to improve their ability to contract for the development of custom designed software.

20 DISTRIBUTION/AVAILABILITY OF ABSTRACT	21 ABSTRACT SECURITY CLASSIFICATION
[X] UNCLASSIFIED/UNLIMITED [ ] SAME AS RPT [ ] DTIC USERS	Unclassified
22a NAME OF RESPONSIBLE INDIVIDUAL: Martin J. McCaffrey	22b TELEPHONE (Include Area code): (408) 646-2488; 22c OFFICE SYMBOL: AS/Mf

DD FORM 1473, JUNE 86 *Previous editions are obsolete* SECURITY CLASSIFICATION OF THIS PAGE

S/N 0102-LF-014-6603 UNCLASSIFIED

Approved for public release; distribution is unlimited

Contracting for Embedded Computer Software
Within the Department of the Navy

by

Henry Attanasio
Major, United States Marine Corps
B.A., Seton Hall University, 1974

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN MANAGEMENT

from the

NAVAL POSTGRADUATE SCHOOL
June 1990

Author: _____
Henry Attanasio

Approved by: _____
Martin J. McCaffrey, Thesis Advisor

_____
Tarek Abdel-Hamid, Second Reader

_____
David R. Whipple, Chairman
Department of Administrative Sciences

ABSTRACT

Over the past 15 years the reliance on computer software, especially within critical weapons systems, has grown in orders of magnitude. Government reports during this same period, however, consistently cite the poor success rate experienced by the Department of Defense in contracting for the development of specialized software as an area of great concern. This thesis investigates the guidance provided by the Department of the Navy, and the contractual methods available, to program managers and contracting officers regarding the procurement of custom designed software. The research conducted concludes that the majority of the blame for software procurement problems should fall upon the personnel within the acquisition system. The problem is a problem of "attitude" concerning software development. Although the system is admittedly cumbersome, the underlying causes resulting in cost overruns, schedule delays, and poor performance are not with the mechanics of the procurement system. Program managers and contracting officers currently have the tools available within the acquisition system to improve their ability to contract for the development of custom designed software.

DTIC Copy Inspected

Accession For
NTIS GRA&I ☑
DTIC TAB ☐
Unannounced ☐
Justification
By
Distribution/
Availability Codes
Dist | Avail and/or Special
A-1

TABLE OF CONTENTS

LIST OF TABLES

I. INTRODUCTION

A. GENERAL

Over the past 15 years the reliance on software, especially within critical weapons systems, has grown in orders of magnitude. Congressional and other governmental reports during this same period, however, consistently cite the lack of success of the Department of Defense in contracting for the development of specialized software. Unfortunately, the flexibility offered by using software as an integral part of a weapon is often stifled by the acquisition system used to procure it. The Department of Defense's ability to contract for the development of this mission critical software directly affects the future defense posture of our Nation.

B. AREA OF RESEARCH AND OBJECTIVE

This thesis examines the contracting and management guidance currently in place regarding the procurement of custom developed software within the Department of the Navy. The object of this research is to explore options available to program managers and contracting officers that may improve the success of acquisition of embedded software.

C. RESEARCH QUESTIONS

1. Primary Question

What currently allowable contractual mechanisms can be used to improve the Department of the Navy's success in the procurement of custom developed software ?

2. Subsidiary Questions

a. How successful is the Department of Defense/Department of the Navy in the procurement of custom software?

b. What directives regulate and/or guide Program Managers and Contracting Officers in the procurement of custom software ?

c. What contract types are typically used for programs involving the development of custom software ?

d. Are there any contractual mechanisms that can be used to aid in the development of custom software ?

D. SCOPE

This thesis investigates the contractual mechanisms and strategies currently allowed by acquisition laws and regulations that may assist program managers and contracting officers in the procurement of mission critical software for weapon systems. The term "mission critical software for weapons systems" refers to embedded computer programs whose failure to perform properly may result in loss of the weapon system, loss of a human being, loss of mission capability, or severe personal injury. Mission critical software excludes all administrative type computer programs.

Although the laws and regulations apply across the board for all Federal agencies, this thesis, in most cases, deals within the confines of the Department of the Navy. It focuses on contractual types, tools, mechanisms and strategies that have been used in the past during software development programs. The results of this research are directed towards Department of the Navy program managers and contracting officers. For the purposes of this research, the success of a program is based on the Department of the Navy's ability to deliver a quality weapon system to the fleet, that is—mission

capable, on time, on budget. Simplifying it one step further by taking a "user-based" approach to quality, success is measured by how a product satisfies the user's needs [Ref. 1] .

This thesis takes a strict managerial view of software development. It does not address specific technical issues. It is the program manager and contracting officer who must bridge the gap between the "ones and zeros" programmers and the end user. With that in mind, this thesis explores the tools available to accomplish such a critical task.

E. METHODOLOGY

1. Literature Search

A literature search was conducted in order to validate the basic premise of the thesis, that in fact there is a problem regarding the procurement of custom developed software by the Department of Defense. Sources of this literature review were General Accounting Office audit reports and the Defense Logistics Studies Information Exchange. Extensive use was made of the Naval Postgraduate School Library, a university library and information center featuring an on-line bibliographic search capability. A review of both government and non-government publications and periodicals was conducted.

2. Survey

A written questionnaire was sent to 150 government and industry professionals with various backgrounds including program management, system engineering, contracting and contract administration, test and evaluation, and quality assurance. A copy of the questionnaire used during the survey is contained in Appendix A. The results of this data collection effort are contained in Chapter V and Appendix B.

3. Interview

Personal and telephone interviews were conducted with government and non-government personnel with varying background experiences. Although the number of interviews conducted was small, an attempt was made to interview a broad spectrum of people involved in the software development process. The areas of expertise of those persons interviewed ranged from corporate vice-president, to contracting officer, to software quality assurance representative, to independent validation and verification personnel. The intent of the interviews was to supplement and amplify that data collected through the survey. A list of the people interviewed is contained in Appendix C.

F. LIMITATIONS

Although 65 of the 150 questionnaires distributed were returned (a response rate of 43%) only 49 had had experience in software development and acquisition. First hand information regarding specific weapon system programs was difficult to obtain. Most of the data concerning the problems encountered during the procurement of embedded mission critical software were obtained through second sources, such as DoD and General Accounting Office (GAO) reports. Reliance on these types of reports, in lieu of data directly from the program offices, may sometimes lead to generalizations of the entire weapon system program based on problem areas in software development. Observations made during this research effort concentrated solely on the software development within a weapon system. A sincere attempt was made to isolate software development problems from any other program problems encountered.

G. ORGANIZATION

Chapter II establishes the background for the problem. Based on a literature review, it discusses the criticality of software development and the government's ability to effectively and efficiently procure that software. The chapter also outlines the Federal laws and regulations regarding software development. Additionally, it contains a brief overview of Department of the Navy's guidance to program managers and contracting officers concerning the acquisition of custom designed software.

Chapter III contains a description of the contract types outline in the Federal Acquisition Regulation (FAR). The program manager and contracting officer must decide which contract type best suits their particular needs. Any of the contract types outlined in this chapter may be deemed suitable for the procurement of custom designed software.

Chapter IV provides an overview of the two methodologies used in software development. The two schools of thought can be summarized into two development philosophies: Design to a firm spec, and the use of rapid prototyping.

Chapter V is a collection of data gathered from the research survey and from personal interviews. It describes the feelings of "the duty experts from the field" with respect to contracting for software development.

Chapter VI draws conclusions from the data gathered and makes recommendations for increasing the success rate of programs involving the development of mission critical software. The chapter concludes with recommendations for future research.

II. BACKGROUND

A. INTRODUCTION

There is virtually no modern convenience today that does not require some piece of software to be functional. From an automatic coffee maker, to a microwave oven, to an automobile, our reliance on software has become phenomenal. Likewise, in order to combat the sophisticated threat, weapons systems have become dependent on software for their mission accomplishment.

In February 1979, the Comptroller General of the United States (GAO) submitted a report to Congress titled, "Contracting For Computer Software Development — Serious Problems Require Management Attention To Avoid Wasting Additional Millions" [Ref. 2]. The report cited contractual and managerial problems from the pre-award phase of the acquisition through contract execution that led to cost overruns, lengthy delays and failures to deliver a usable product. The GAO investigated nine software development programs in detail. Eight of the nine programs were classified as "failures". Only one of the nine programs investigated produced software that was capable of being used as delivered. The major contractual problems cited in the report are listed below:

(1) Federal agencies contract for software development with little specific guidance.

(2) Agencies also overestimate the stage of system development they have reached before they contract.

(3) The lack of a good contractual description of what the contractor is to do makes it difficult for the agency to claim poor contractor performance.

(4) Agencies quickly over commit themselves and fail to control contractors through strict phasing.

(5) Agencies do not manage software development contracts during execution.

(6) Agencies accept and pay for software without adequately inspecting and testing.

(7) Agencies fail to establish a single focal point for communication with contractors.

(8) Agencies do not adequately specify or enforce contract clauses for recovery in the event of poor performance.

In 1989, a staff study entitled, "Bugs In The Program, Problems In Federal Government Computer Software Development And Regulation" [Ref. 3], was submitted by the Subcommittee on Investigation and Oversight to the Committee on Science, Space and Technology, U.S. House of Representatives. The study cited that,

> Getting value for the Government's money is a recurring problem. Computer software, which is now a major cost item in many procurements, is not immune from traditional procurement problems such as delay, cost overruns and poor performance... Worse, the procurement system as presently structured does not take into account the special needs of computer software systems and can compromise effective software development from the start.

The report further states that:

> ...software cannot be properly developed using the welter of regulations presently in force. While software now drives system requirements, the procurement system still focuses attention on hardware...

Today's weapon systems have grown tremendously more complex. Their reliance on embedded software has increased in orders of magnitude. Examples of this reliance on software are shown in Table I [Ref. 4], comparing an early 60's weapon system with those of tomorrow. Tables II and III [Ref. 5] illustrate the cost trends and the increase in the Department of Defense's

TABLE I. Weapon Systems Software Complexity Comparison

WEAPON	LINES OF SOFTWARE CODE*
F-4	0 (virtually)
F-16D	236,000
C-17	750,000
B-1B	1.2 million
ATF	5 - 7 million
SDI	25 million (estimate)

* Lines of code are often used to describe the complexity of a software program. It should also be noted that a doubling of the lines of code does not necessarily equate to a doubling of complexity, and more likely results in a program 10 times more complex.

Source: Kitfield, James. "Is Software DoD's Achilles' Heel?" *Military Forum*

TABLE II. Cost Trends: Hardware Versus Software
(percentage of total costs)

	1955	1970	1979	1985 (estimate)
COMPUTER HARDWARE	83	45	25	10
SOFTWARE	7	55	75	90

Source: Glaseman, Steven. *Comparative Studies in Software Acquisition*

TABLE III. Software Investment by DOD (billions of dollars)

	1979	1985
TOTAL DOD SOFTWARE	7	12
EMBEDDED SOFTWARE	5.25	10+

Source: Glaseman, Steven. *Comparative Studies in Software Acquisition*

expenditures on software, respectively, further amplifying this growing reliance on software within our major weapons systems. DOD investment in mission critical software is predicted to be $30 billion in 1990 [Ref. 6]. As this dependence on software increased, industry has responded by dedicating more of its program resources to software development. Table IV [Ref. 5] shows an example of this required increase by the McDonnell Douglas Aircraft Company.

TABLE IV. McDonnell Douglas Resource Assignment
(percentage of development personnel)

	1960 (F-4)	1970 (F-15)	1980 (F-18)
NONCOMPUTER HARDWARE	98	74	57
EMBEDDED COMPUTER HARDWARE	1	2	3
EMBEDDED COMPUTER SOFTWARE	1	24	40

Source: Glaseman, Steven. *Comparative Studies in Software Acquisition*

Unfortunately, the increased reliance on embedded software was not met with an increased emphasis on the management of software development on the part of both the government and industry. Over the past 15 years, contractors have had little incentive for providing a quality product. This does not presume there was an intent to defraud, but rather, that from a managerial aspect there existed severe problems on both sides. The government failed to use its contractual "tools/power" to identify standards of performance, track development programs, adequately test deliverables, and "hammer" nonperformance. The only thing that a contractor could count on seems to have been the fact that follow-on contracts for software maintenance and modification would somewhat reward the delivery of a non-quality product [Ref. 7]. As we have evolved into a data processing/software dependent environment, both parties are now well aware of the criticality of delivering a quality product to the end user. The fact remains, however, that the management of a software development program can be a "nightmare" [Ref. 8].

Various reports at all governmental levels recognize that the requirements of today's rigid procurement system contrast with the flexibility afforded by the current methodologies used in software development and engineering. In other words, the tremendous flexibility offered by using software is often stifled by the acquisition system used to procure it. With today's shrinking defense budget, the program manager and contracting officer must team up to formulate an acquisition strategy that will be both workable and effective. Although a change in procurement methodology regarding software development is often recommended, in the short term this is unlikely to occur. Today's program managers and contracting officers must formulate their procurement strategies within the existing laws, regulations, and guidelines to successfully develop and field the weapon systems of tomorrow.

B. CURRENT ENVIRONMENT

It has been eleven years since that first General Accounting Office report was issued and we still hear the same exact story. Time after time, there are system development, testing, production, and fielding delays, some of which lead to entire program cancellations. These types of results in the development of custom software for the Department of Defense, either for multipurpose use or embedded within a major weapon system, seems to be the norm rather than the exception.

Recent reports indicate that software development problems are a major factor in the current schedule delays and the estimated cost overruns as high as $500 million on the Air Force's C-17 transport program. Four years after the deployment of the B1-B, mission critical software discrepancies still exist. Eight years and $346 million were spent by the Air Force in trying to develop the worldwide military command-and -control information system (WIS). The program was renamed and reassigned to the Defense Communications Agency after persistent software development problems. [Ref 4]

The command and control center for the North American Aerospace Defense Command (NORAD) and the U.S. Space Command is the Cheyenne Mountain Air Force Station. There, data processing equipment supporting the command's tactical warning and attack assessment (TW/AA) mission was scheduled for replacement in 1986. The Air Force invested approximately $72 million in the initial software upgrade to the Semi-automated Technical Control Unit. In December of 1987, formal qualification testing of the initial software upgrade was completed. Originally scheduled for a 2-month period, qualification testing was conducted over 10 months. The software was considered "unstable", a term applied to software that is unpredictable, and may not

produce consistent results when run against a known set of operating conditions. [Ref. 9]

The sister system to the one described above is the Air Force's Space Defense Operations Center (SPADOC) also at the Cheyenne Mountain complex. It too has had some major problems. After a $235 million investment in the system and a four year schedule slip, SPADOC could only meet seven of the 23 mission functions stated in the contract. Last year's estimates showed a program cost growth of over 65% to $437 million with the original operating date of June 1988 being slid until sometime in fiscal year 1994. [Ref. 10]

Recently, the Army announced a 26 month slip in IOC for their Advanced Field Artillery Tactical Data System, an integral part of their Tactical Command and Control System. The number one cause for the delay, cited in a GAO report, was "software development and reliability problems". [Ref. 11] Between the program's approval in 1986, and a similar GAO report to Congress in December 1989, cost to complete estimates had risen by $600 million [Ref. 12]. The Navy's next generation attack submarine, the SSN-21 Seawolf, will not be fully mission capable when delivered in 1993. It's combat system, the AN/BSY-2, will not be fully operational because, "it will lack certain necessary software". [Ref. 13]

No Service is exempt from their share of the problems. The Marine Corps cancelled the Marine Integrated Fire and Air Support System (MIFASS) program after a 12 year development effort at a cost in excess of $200 million. One of the major reasons for cancellation of the program was the government's inability to manage the development of the computer software required to meet the MIFASS mission need [Ref. 14,15].

General Bernard Randolf, chief of the Air Force Systems Command, summed up the government's track record in software acquisition, "On

software schedules, we've got a perfect record: We haven't met one yet" [Ref. 4: 29].

C. LEGISLATIVE ACTION

There are two legislative actions that affect the procurement of automated data processing (ADP) equipment, to include computer software. The first was an amendment to The Federal Property and Administrative Services Act of 1949, and is commonly referred to as the Brooks Bill (PL 89-306). Enacted by Congress in 1965, this bill established the authority and procedure for consolidating all ADP activities such as purchasing, leasing, equipment transfer, servicing, and maintenance. It made the General Services Administration (GSA) responsible for coordinating all ADP procurement activities in the Government. Federal agencies no longer had the authority to determine their own ADP policies. [Ref. 16,17]

Congress passed the second legislative action regarding ADP procurement in 1981 as an amendment to the Fiscal Year 1982 Appropriations Act (PL 97-86). Known as the Warner Amendment, this law excluded the Department of Defense from the provisions of the Brooks Bill when the ADP equipment:

> ...involves intelligence activities; involves cryptologic activities related to national security; involves the command and control of military forces; involves equipment that is an integral part of a weapon or weapon system; or subject to subsection (b), is critical to the direct fulfillment of military or intelligence missions.

Subsection (b) specifically states that routine administrative and business ADP equipment is not excluded. [Ref. 18]

D. ACQUISITION REGULATIONS

There are two major acquisition regulations that govern Department of Defense procurement efforts. They are the Federal Acquisition Regulation (FAR) and the Department of Defense Federal Acquisition Regulation Supplement, the DFARS. While the FAR provides thousands of pages of regulatory guidance to the contracting officer and the program manager, it does not differentiate between contracting for software development and contracting for any other goods or services. The FAR deals with the mechanics of contracting vice strategy formulation.

Part 270 of the DFARS addresses the acquisition of computer resources. The majority of information provided in this section, however, is merely a reiteration of the legislative requirements governing the procurement of ADPE. Other than a one page description of the Warner Amendment exclusions to the Brooks Bill, it does not address the procurement of custom designed or embedded software. [Ref. 19]

E. DEPARTMENT OF DEFENSE PROCUREMENT GUIDANCE

The Department of Defense has published various documents issued as either directives or instructions regarding the department's procurement programs. Keeping in mind the basis for this research is managerial in nature, the major acquisition directives are discussed below. Many other directives specifically addressing software development issues have been promulgated. They, however, deal with specific technical requirements rather than overall acquisition strategies and will, therefore, not be addressed.

1. DOD Directive 5000.1

Department of Defense Directive 5000.1, dated 1 September 1987, and titled, Major and Non-Major Defense Acquisition Programs, provides

guidance to the Services regarding the classification of procurements. DOD procurements are classified as major programs if their cost will exceed $1 billion in total procurement funding or if their total research, development, test and evaluation (RDT&E) funding will exceed $200 million. There are no direct references to software development programs. The two threshold criteria, as well as the other provisions of the directive, would apply to programs involving software development. There are other criteria by which a program can be designated as a major acquisition, such as the amount of Congressional interest. [Ref. 20]

2. DOD Instruction 5000.2

"Defense Acquisition Program Procedures," DOD Instruction 5000.2, establishes uniform procedures regarding the procurement of major programs. This document mostly discusses the approval process through the Defense Acquisition Board (DAB) and the necessary paperwork trail required to document the program. It, again, does not specifically address any of the aspects of software development within the acquisition cycle. [Ref. 21]

3. DOD Directive 5000.3

The first major publication that delineates guidance regarding software development is DOD Directive 5000.3, Test and Evaluation (T&E). This directive establishes specific policy concerning the test and evaluation of computer software. It requires that the, "principles and methodologies provided in DOD 5000.3-Manual-3 be applied to the T&E of system computer software." The manual itself provides guidance for addressing software in the Test and Evaluation Master Plan. [Ref. 22]

4. DOD Directive 5000.29

DOD Directive 5000.29, Management of Computer Resources in Major Defense Systems, "establishes the policy for the management and

control of computer resources (hardware and software) during the development, acquisition, deployment, and support of major defense systems." This directive requires programs to validate mission requirements, conduct appropriate risk analyses, and provide for complete planning throughout the computer resource life cycle. [Ref. 23]

F. DEPARTMENT OF THE NAVY PROCUREMENT DIRECTIVES

The majority of the Department of the Navy's procurement guidance implements the Federal and Department of Defense acquisition requirements within the Navy. Specific directives regarding the development of software begins to focus more on the Navy's philosophy for the management of mission critical computer resources.

1. SECNAVINST 5200.32

This instruction, "Management of ECR in the Department of the Navy," implements DOD guidance for the management of computer resources in defense systems within the Department of the Navy. It also promulgates additional policies and procedures for the management of Navy weapons, communications, command, control and intelligence systems when embedded. [Ref. 24]

2. SECNAVINST 5200.37

The Secretary of the Navy Instruction 5200.37 entitled, "Acquisition of Software-Intensive C2 Information Systems," delineates acquisition policy for software-intensive command and control information systems. This instruction promotes, "routine user involvement in software development process... encourages rapid fielding of needed capabilities." [Ref. 25]

3. OPNAVINST 5200.28

This instruction, "Life Cycle Management of Mission Critical Computer Resources for Navy Systems Managed Under the Research, Development and Acquisition Process," amplifies SECNAV and OPNAV policies for the acquisition, management, and life cycle support of software and related computer resources. Provisions of the Standard Embedded Computer Resources Review Program are also implemented within the Navy. [Ref. 25]

4. MCO P5000.10C

Although Marine Corps Order P5000.10C, the "Systems Acquisition Management Manual," states its purpose is to, "publish management guidance and procedures...for the acquisition of weapon systems, computer resources, and equipment..." it makes little reference to software development. The manual refers to software development in a single subparagraph in a brief overview of Full Scale Development (FSD). It does not give the program manager sufficient guidance regarding mission critical computer resources (MCCR). The order does refer the reader to MCO 5200.23 to find the policy and procedures for the management of MCCR. [Ref. 26]

5. MCO 5200.23A

Marine Corps Order 5200.23A, "Management of Mission Critical Computer Resources (MCCR) in the Marine Corps," implements the policies and guidance promulgated by the Department of Defense and the Secretary of the Navy. It establishes a Marine Corps policy for the acquisition and management of MCCR. The order states its purpose as, "to reduce the life cycle support costs of all Marine Corps mission critical systems by reducing the proliferation of mission critical system hardware and software." [Ref. 27]

6. Naval Systems Command Instructions

Various instructions and guidance have been promulgated by each of the Navy's Systems Commands. The Naval Air Systems Command (NAVAIR), the Naval Sea Systems Command (NAVSEA), and the Space and Naval Warfare Systems Command (SPAWAR) have all issued their own program manager's policy guidance regarding software development. Most of these instructions direct the implementation of DOD guidance and assign areas of responsibility within the command. [Ref. 25]

7. Tactical Digital Standards (TADSTAND)

A series of policy documents have been developed by the Navy regarding system software development and support. Published as Tactical Digital Standards (TADSTAND's), they provide guidance for the standardization of such things as; embedded computers, computer peripherals, input/output interfaces, standardized languages, software testing and documentation requirements for the Navy's mission critical systems. TADSTAND's serve to both mandate the use of certain DOD Standards in software development and testing, and also identify the Navy's unique mission critical computer resources requirements. [Ref. 8,25]

G. SPECIFICATIONS AND STANDARDS

The Department of Defense publishes a series of reference documents which can be used by all parties in military procurement. These documents fall into three categories; Military Handbooks, Military Specifications and Military Standards. Handbooks bring together procedural and technical or design information related to components, processes or services. Specifications are complete and detailed descriptions of products which are either strictly military in nature, or are modified commercial products

requiring special features to satisfy military mission needs. Standards describe engineering and management processes, methods and design criteria, testing techniques, and data collection requirements. It is important to remember that handbooks, specifications, and standards are merely guides to assist the government's program manager and contracting officer. They are not laws. Once placed in a contract, however, they become legally binding. The major specifications and standards regarding software development are outlined in this section.

1. DOD-STD-1467

Although primarily written as an Army document, "Military Standard Software Support Environment," is available for use by all Departments of DOD. This document provides the uniform minimum requirements for the contractor to define a Development Software Support Environment (DSSE), to ensure compatibility with the government's designated Life Cycle Software Support Environment (LCSSE). In other words, the program manager, the contracting officer, and the contractor must plan for the life cycle support of deliverable software throughout the development cycle. This makes all three parties not only aware, but also, responsible for software support after the software becomes operational. The standard is written in such a way as to not dictate a specific approach, but to allow the software contractor, "the flexibility to develop software and manage the contract in accordance with the contractor's best judgment and practices." [Ref. 28]

2. DOD-STD-1703

This standard, "Software Product Standards," looks at the software development process more from a manager's perspective. It is the best document to use as a guide in the formulation of an acquisition strategy for a software development program. It provides the program/project manager

valuable guidance, without giving the impression that the recommendations provided are mandated. DOD-STD-1703 encourages the manager to tailor program requirements, up-front, placing emphasis on the requirement for proper documentation of the contractor's efforts, rather than attempting to control or direct those efforts. Table V provides some basic questions that the program manager must answer when tailoring program requirements.

TABLE V. Factors that Affect Program Requirements

1.	How large and how complex is the software product ?
2.	Who will use the software product ?
3	How long will it be used ?
4.	Will someone be required to provide life-cycle support for the product after it is developed ?
5.	During development, which of the following does the customer consider most important: cost control, schedule control, or functional capability ?
6.	What are the development risks ?
7.	How large is the development staff ?

Source: DOD-STD-1703

Table VI provides, "the axioms of software development," some very basic and simple guidance, but at the same time some very valid and valuable rules to live by. [Ref. 29]

TABLE VI. Axioms of Software Development

1. To be successful, a software project must be well-managed.
2. Good management requires planning, attention to detail, and discipline.
3. Documentation is a primary vehicle with which managers manage.
4. Until a software product is tested, no one will know its quality.
5. Both customers and users have a vital role to play in any software development effort; documentation gives them the information necessary to perform their role.

Source: DOD-STD-1703

3. DOD-STD-2167A

Published in 1988, the standard for "Defense Systems Software Development," is the latest guidance promulgated by the DOD. It establishes uniform standards for software development that are applicable throughout the system's life cycle. DOD-STD-2167A establishes the requirements, however, it does not specify how the contractor is to meet those requirements. For example, the requirement to have a software management structure in place is worded as follows; "...the contractor shall implement a process for managing the development of the deliverable software." Again, emphasis is placed on the program manager's ability to determine his particular needs, and tailor his project requirements accordingly. All aspects of software design, development and testing are addressed in this standard. [Ref. 30]

4. DOD-STD-2168

DOD Standard 2168, "Defense System Software Quality Program," contains the requirements for the development, documentation and implementation of a software quality program. It interprets the applicable quality requirements set forth in MIL-Q-9858, the most stringent of the quality assurance specifications, for software. Published in 1988, it supersedes the long standing software quality assurance guidance found in MIL-S-52799A. [Ref. 31]

5. MIL-HDBK-286

This handbook, "Software Quality Evaluation," is currently available only as a draft, and is intended to be used with DOD-STD-2168. It provides guidelines for applying the requirements of the Department's Software Quality Program. [Ref. 32]

6. MIL-HDBK-287

Military Handbook-287, published in 1989, is "A Tailoring Guide for DOD-STD-2167A, Defense System Software Development." It provides just that, guidance to the program managers and other program office personnel for tailoring the requirements of 2167 for a software development or support contract. [Ref. 33]

7. MIL-HDBK-782

This handbook, "Software Support Environment Acquisition," was written to provide personnel not familiar with the requirements of DOD-STD-1467 the fundamentals required to ensure supportability on contracted software development programs. Emphasis is placed on the concern for software supportability beyond the contractual delivery date and its subsequent operational use. It is intended for use by all personnel in the procurement process responsible for ensuring life cycle software supportability. [Ref. 34]

H. MANAGEMENT GUIDANCE

There are many documents that are readily available to assist the Navy/Marine Corps program manager and his staff through the various phases of an acquisition. Although these documents are informal, that is, they are not published through the Navy Directives Issuance System, they often provide the insight necessary for the program manager to perform successfully. Three examples that would be pertinent to the acquisition of computer software are cited below.

1. RDT&E/Acquisition Management Guide

The Navy's, "Research Development Test & Evaluation/Acquisition Management Guide – 11th Edition," published in January of 1989 misses the mark when it comes to assisting the program manager with software development issues. There is a only one paragraph addressing mission critical computer resources. Unfortunately, it does not address development or testing requirements, but rather, the requirement for an approved Computer Resources Life Cycle Management Plan (CRLCMP) prior to the Milestone II decision point. It says in part:

> Advanced, fully integrated weapons, avionics, intelligence, and command, control and communications technologies are gaining increasing importance in Navy and inter-service weapons systems. The nuclei of such integrated systems are embedded Mission Critical Computer Resources (MCCR)...[Ref. 35: 2-19]

Although the document seemingly recognizes the "increased importance" of embedded software, it fails to provide any further guidance. [Ref. 35]

2. Acquisition Strategy Guide

In 1984, the Defense Systems Management College, Fort Belvoir, Virginia published the "Acquisition Strategy Guide." Its purpose was to provide, "information that Program Managers should find useful in structuring,

developing, and executing an acquisition strategy." Like the RDT&E manual, this publication addresses the mechanical aspects of strategy formulation. It fails to identify any unique strategy formulation requirements when dealing with the development of custom designed, embedded software. It does not recognize the difference between hardware and software; it merely addresses programs as a whole. [Ref. 36]

3. Navy Program Manager's Guide

The "Navy Program Manager's Guide" is the first publication found that places the importance of software development management in perspective:

> The combat launch of an aircraft without its missiles or the sortie of a submarine without its torpedoes makes successful destruction of enemy forces highly unlikely. To any professional, this fact is obvious.... Far fewer, however, know that a weapon systems with full armament installed may be equally failure-prone because of embedded computer resource (ECR) deficiencies. ...PM's universally have become aware that we can no longer fly, dive, steam or fight without ECRs in most of our weapons. We use ECRs to make our systems operate, to test them, to produce them, to adapt them, and to keep them responsive to changing threats. [Ref. 8: 4-67]

The manager is made aware of the critical role that embedded computer resources, and their management, play in mission accomplishment. The guide provides the program manager a "heads-up" stating, "Software management has been recognized by successful PM's as requiring early, intensive, continuing management." It goes on to provide excellent guidance in general terms and recommends various sources of assistance available within the Navy. [Ref. 8]

I. SUMMARY

This chapter has provided the background for the complex environment in which program managers and contracting officers must operate when

procuring embedded software. The software development costs associated with major weapon system programs can be astronomical. Armed with very little formal guidance these "procurement experts" must operate within the law to ensure a quality product is delivered on time and on budget. When it comes to contracting for custom software the program manager and contracting officer are actually afforded alot of flexibility. They must rely heavily on their common sense approach and good business judgment. Problem programs quickly come under the scrutiny of the Congress, the GAO, the IG's, and various auditors. Chapter III will discuss the specific types of contracts available to the program manager and contracting officer for the procurement of embedded software.

III. CONTRACTING FOR SOFTWARE DEVELOPMENT

A. INTRODUCTION

The single most important document in the procurement of custom designed software is the contract itself. The contract acts as the legal basis for performance, and the importance of stating in the contract what, in fact, is intended can not be overemphasized. The government and the contractor must collectively enter into this agreement, understanding that it is a mutual effort with both parties having their own responsibilities. Various types of contracts are outlined in the Federal Acquisition Regulation (FAR). The first consideration in choosing a contract type is usually that of risk assumption. The spectrum of contract types range from a Cost-Plus-Fixed-Fee type contract in which the government assumes the majority of the risk, to a Firm-Fixed-Price type contract which places the greatest risk on the contractor. This chapter will provide a brief overview of the types of contracts that may be used for the acquisition of custom developed software.

B. COST REIMBURSABLE CONTRACTS

A cost reimbursable contract is one that allows the contractor to receive payment for all costs that are deemed "allowable and allocable". These type contracts do not contain a cost ceiling, thus the government assumes the majority of the risk. The contractor provides the government with an estimate to complete the task, however, the contractor is not bound by that estimate. It is possible to expend all of the funds allocated for a particular program and have a situation where the contractor is not legally bound to

produce a deliverable. The following are variations of this cost reimbursable concept.

1. Cost-Plus-Fixed-Fee

A cost-plus-fixed-fee (CPFF) contract is a cost reimbursement contract that establishes a fixed fee payment to the contractor. The contract price is based on a negotiated cost to complete the required task plus a negotiated contractor's fee. The contractor is not legally bound by this negotiated cost figure. As implied by the contract name, the contractor's fee remains fixed regardless of total contract price to the government. The contractor receives the negotiated fee regardless of the actual costs incurred. At the same time, the contractor has a legal claim for his actual costs. Under this arrangement the government assumes the majority of the risk, and the contractor has little incentive to control costs. [Ref. 37: 16.306]

2. Cost-Plus-Incentive-Fee

In a cost-plus-incentive-fee (CPIF) contract the government and the contractor agree to a target cost based on the required task. Like the CPFF contract, the contractor has the right to be reimbursed for all allowable expenses incurred. The difference, however, is that the contractor's fee is adjusted by using a formula that compares the total allowable costs to the negotiated target cost. Thus, the contractor has the incentive to effectively manage cost control. At the onset of the effort, a target cost, target fee, a minimum and maximum fee, and an adjustment formula are negotiated. If the contractor completes the task below the target cost the fee is adjusted upward based on a cost share ratio established in the contract. Likewise, if the contractor's actual costs are over the target cost the fee will be adjusted downward. The minimum and maximum fee figures are included in the contract, on the one hand, to protect the contractor from being in the position of having

to work for no fee, and on the other, to prevent the contractor from taking cost short cuts to merely increase his profit. This type contract is best suited for a development effort when a realistic target fee and target cost can be negotiated that will effectively incentivize the contractor to control costs. [Ref. 37: 16.404-1]

3. Cost-Plus-Award-Fee

A cost-plus-award-fee (CPAF) contract is also a reimbursable type contract. Under this type arrangement the contractor's fee is divided into two parts. The first consists of a base fee established at the onset of the contract, and the second, consists of an award amount or pool that is set aside. The contractor may earn all or part of this award fee pool based on the government's "judgmental evaluation" of the contractor's performance. The intent is to monetarily reward the contractor for performance above the minimally accepted standard set at the beginning of the contract. Like other incentive type contracts, the contractor can be rewarded for technical performance, supportability and reliability achievements, or superior quality. A service contract to run a military dining facility can be used as a simple example. An award pool could be structured to incentivize the quality of the contractor's food, service, and cleanliness requirements beyond that which was established as the minimum acceptable standard in the contract. A government appointed person or committee periodically awards portions of the monetary pool based on their subjective impressions of the contractor's performance. [Ref. 37: 16.404-2]

C. FIXED PRICE CONTRACTS

In a fixed price type contract the contractor is legally bound to deliver the goods or services for a predetermined price regardless of his actual costs.

This type contract is most appropriate in a production type buy where product designs have been stabilized. "Key features of a fixed-price contract are: contractor promises to deliver on time, per specification, for a fixed price, and the government promises to pay the fixed price if the product/service conforms to the contract" [Ref. 38]. In this type agreement, the contractor assumes majority of the cost risk. There several variations of the fixed price type contract listed in Part 16 of the FAR. They are listed below:

- Firm-Fixed-Price (FFP)
- Fixed-Price with Economic Price Adjustment (FPE)
- Fixed-Price Incentive, Firm Target (FPIF)
- Fixed-Price Incentive, Successive Targets (FPIS)
- Fixed-Price with Prospective Price Redetermination (FPRP)
- Fixed-Price Retroactive Price Redetermination (FPRR)
- Firm-Fixed-Price, Level of Effort (FFP, LOE)

When contracting for software development, typically used contract types can be placed in four general categories. They are discussed below.

1. Firm-Fixed-Price

The simplest of any contracts is the firm-fixed-price agreement. In this agreement, the contractor is paid the contract price upon acceptance of the work. There are no adjustments to total price regardless of the actual cost to perform. The contractor's profit or loss posture has no bearing on the price paid once the contract is awarded. This arrangement, "provides maximum incentive for the contractor to control costs and perform effectively and impose a minimum administrative burden upon the contracting parties" [Ref. 37: 16.202-1].

2. Fixed-Price-Incentive

Fixed-price-incentive contracts have one of two options. They will either have an established Firm Target (FPIF) or a series of Successive Targets (FPIS). Both of these arrangements provide for an adjustment to the contractor's profit based on the relationship between the final negotiated total contract cost and the target cost established at contract award. At the onset of the contract, a target cost, target profit, ceiling price, and a share ratio for establishing the final profit and price are negotiated. At the completion of the contract a final contract price is determined, and the contractor's profit posture reflects the application of this negotiated share formula. When the final cost is more than the target cost, the contractor's profit will be less than the target profit, and in some cases may even result in a net loss. With profit tied to this share ratio, the contractor is incentivized to control costs. The ceiling is established to protect the government from paying more than that total price for the contract. Unlike the cost reimbursable contracts, the contractor must satisfactorily complete the contract task in the form of a service or deliverable regardless of cost. The only difference between the two FPI contracts is the time at which the share formula is applied. In a successive target situation, the share ratio is computed at different times throughout a production effort. The FPIS contract would be appropriate in a situation where the cost or pricing data were not adequate at the beginning of the production run to negotiate a firm-fixed-price contract. [Ref. 37: 16.403]

3. Fixed-Price with Economic Adjustment

When the contracting officer determines that the market conditions are so unstable that the either government or the contractor need to be afforded some level of protection, an economic adjustment type contract is

appropriate. Section 16.203-1 of the FAR identifies three contingencies for price adjustments:

(1) Adjustments based on established prices.

(2) Adjustments based on actual costs of labor or material.

(3) Adjustments based on cost indexes of labor or material.

The upward or downward adjustments may be up to ten percent of the contract's base price. The price adjustment clause in the contract lists specific contingencies upon which the price adjustment will take effect. Other than these delineated events the contract is executed virtually the same as a firm-fixed-price contract. This type of contract would not normally be used in a software development effort, it is however possible, and is included for purposes of this discussion. [Ref. 37: 16.203]

4. Level of Effort

Typically used for a feasibility type study in a research or development effort, the contract establishes a firm-fixed-price for a specified level of effort rather than an end product. Payment is based on the level of work expended and not the achieved results. The scope of the work is loosely defined allowing the contractor the greatest flexibility. For example, the government might hire a contractor to perform 1000 man-hours of research. The contractor assumes almost no financial risk, in that, he will get paid for expending those 1000 hours regardless of the outcome. [Ref. 37: 16.207]

D. OTHER CONTRACT OPTIONS

The FAR also states that contracts, "may be of any type or combination of types that will promote the Government's interest..." [Ref. 37 16.102.b] Gerald Lee Francom, while a student at the Naval Postgraduate School, explored the feasibility of using a Fixed-Price-Award-Fee contract. He

combined the advantages of the Fixed-Price contract with the incentive provisions of an Award Fee contract and proposed that this type contract be recognized within the FAR. [Ref. 39]

The Air Force has in the past used this combination of contract types with great success. The improvement program for the B-1B bomber Computer Integrated Test System (CITS) was awarded under a Fixed Price Incentive Firm with an Award Fee (FPIF/AF). The original contract specification called for a two percent or less false-alarm rate which was met by the contractor. After the system was delivered this government specified rate was deemed to be inadequate by the end users. Thus, a program to reduce the number of CITS false-alarms was undertaken. Under the FPIF/AF pricing arrangement, the contractor was incentivized to provide high management emphasis on quality, timeliness, and cost-effectiveness during this effort. The resultant program produced a false-alarm rate that was below .03 percent. In this case, the incentive contract, "resulted in a true win/win situation for both the government and the contractor." [Ref. 40]

E. CHOOSING THE PROPER CONTRACT

"Selecting the contract type is generally a matter for negotiation and requires the exercise of sound judgment" [Ref. 37: 16.103]. Contracting officers are given the latitude, in most cases, to choose the type of contract that best suits their particular needs. As previously stated, the gaiting factor in this decision is usually the question of "risk assumption". The two risks that are analyzed prior to choosing a contract type are cost risk and technical risk. A third risk in the formula is schedule, however, schedule risk is mostly a function of the other two. Dr. Harvey J. Gordon, in an article discussing the role of the contract stated:

> Not all major weapon system contracts are designed to accomplish precisely the same purpose. There are, however, invariants that thread a common course. Despite differences in contracting technique, we strive always for the lowest possible cost, timely delivery, and maximum technical accomplishment within cost and schedule constraints. [Ref. 41: 30]

Every program manager and contracting officer attempts to accomplish this not so easy task. An accurate assessment and subsequent assignment or assumption of risk is a critical factor in achieving this goal. Selection of the proper type of contract assigns cost, schedule and technical risk to the appropriate party.

F. SUMMARY

This chapter provided an overview of the specific contract types outlined in the Federal Acquisition Regulation. It also showed that the contracting officer is given the flexibility to choose the contract type that bests suits the program's needs. The ultimate goal is to achieve a win/win situation using the appropriate contract type. The government will receive a quality produce/ service and the contractor will earn a fair and reasonable profit.

IV. SOFTWARE DEVELOPMENT

A. SOFTWARE DEVELOPMENT PHILOSOPHY

The classical approach to software development, known as the "waterfall model," is illustrated in Figure 4.1 [Ref. 42]. This approach is quite compatible with the government's current contracting philosophy. Using this classical software development method, requirements are defined "up-front", similar to the acquisition

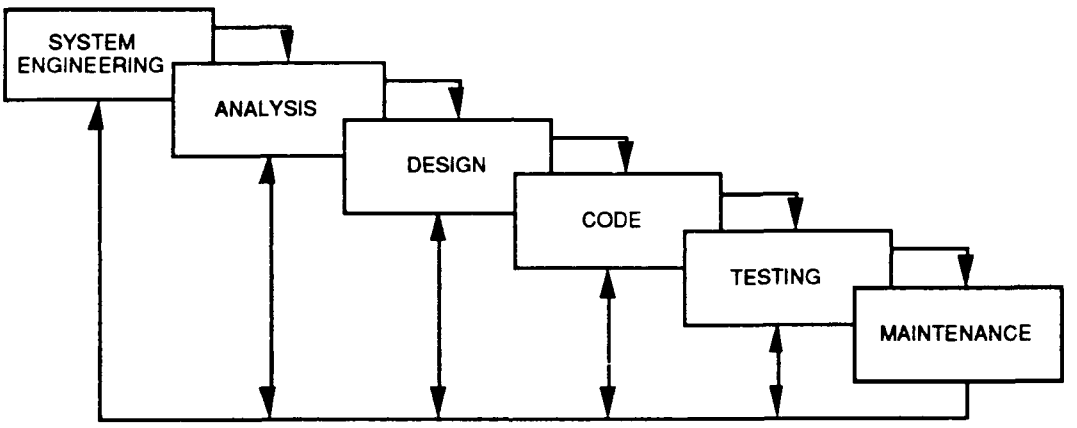


Figure 4-1. Classic Software Life Cycle

process of publishing a detailed specification or statement of work. A contract is awarded delineating the specific design criteria in detail. The contractor designs and tests in accordance with the contractual specifications. Very little flexibility is afforded to either the contractor or the government once these requirements are integrated into the development contract. System development, especially in the case of complex weapon systems, very rarely follows

the clearly defined sequential waterfall model. In the real world, users do not easily communicate or sometimes even know their true requirements. Contracting officers do not easily translate user requirements into definitized contracts. Software engineers do, in fact, need the flexibility during design to accommodate the inherent unknowns in software development.

The more modern approach to development, illustrated in Figure 4.2 [Ref. 42], uses what is known as "rapid prototyping." Using this concept, the user will define a group of objectives for the software to be built. This quick

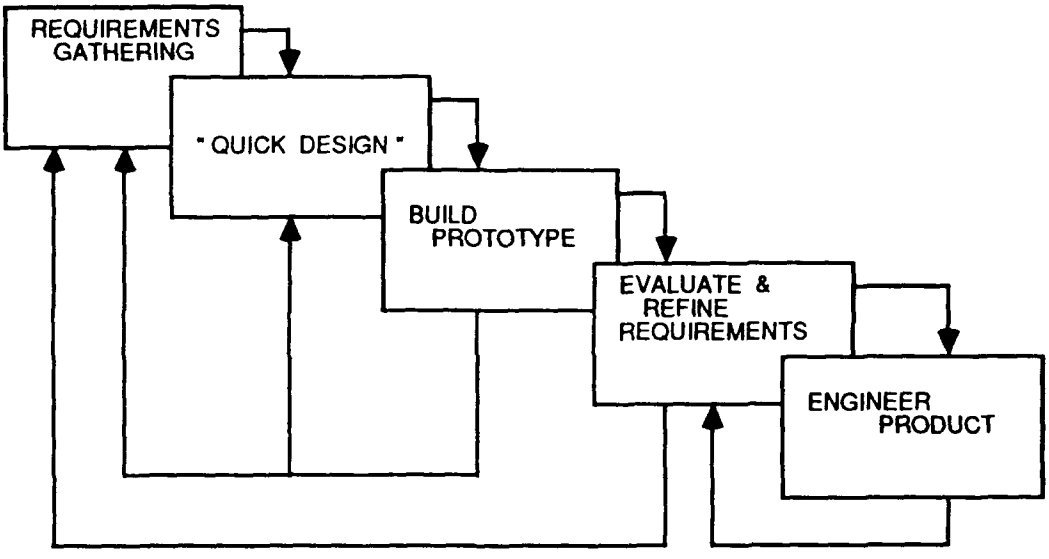


Figure 4-2. Prototyping

design concentrates on generalized functions of the software, paying particular attention to those aspects of the program which are visible to the user. The quick design evolves into a prototype. The prototype then serves as the tool to identify the user's true requirements. It is probably safe to assume that in the majority of design efforts the user does not have a real handle on what is required, at least to the level of detail that would be necessary to

begin full scale software development. Initial concepts will most likely change. The impact of this change will vary depending on the stage of the program. Dr. Roger S. Pressman's comparison of the cost of change is illustrated in Figure 4.3 [Ref. 42: 17]. The use of an iterative process through prototyping allows maximum interface between the ultimate user and the developer. Prototyping prior to full development accommodates this inevitable change. It does so at the easiest and least costly point in the software life cycle. Thus, the user better identifies his requirements and the developer better understands those requirements.

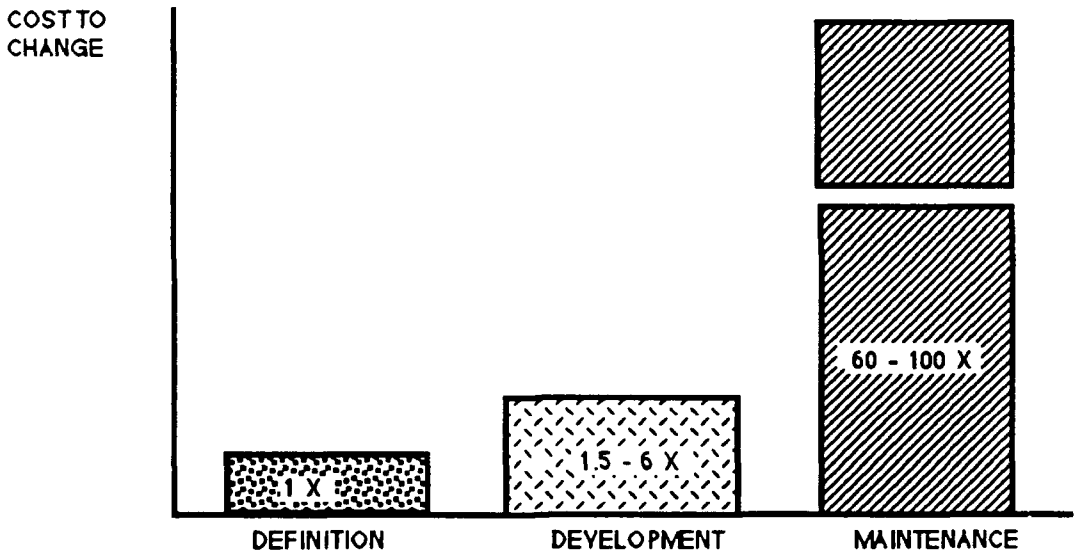


Figure 4.3. Cost impact of Change

The program manager's ability to create this environment where the end user plays an integral role in the early stages of development is one of the essential elements in a successful program. Afterall, who better knows the true mission requirements than the end user. On the other hand, extreme

care must be taken to not allow the user to "gold plate" the requirements, just because the software capability exists. If the requirement is to track ten inbound airborne targets simultaneously, then that should be the design criteria. The attitude that it "sure would be nice" to track 100 inbound targets should be stifled if the tactical mission and/or the hardware will not support that requirement. The fact that the software can "easily" accomplish this task is not relevant. Respondents to this thesis' survey overwhelmingly cited the inability to identify the user's requirements as the number one problem in software development today. Here, the program manager and contracting officer assume the role of mediator.

The prototyping philosophy seems to be accepted by industry (and, as a matter of fact, the government) as the best approach to a new software development program. Unfortunately, there exists an inherent conflict between this approach and the today's acquisition system. The government's procurement system operates on the premise that all work will be specified in detail before the contract is awarded. The prototype approach uses broad functional descriptions to loosely define the end product's objectives, not the work to be performed, ergo, the conflict. The program manger and contracting officer's ultimate challenge is to choose the best design philosophy and fit it into a contractual mechanism.

B. SOFTWARE DEVELOPMENT ENVIRONMENT

As program managers and contracting officers, on both sides, become more software literate, the ability to contract for software development should become somewhat easier. At least to the extent that all players in the contracting process will have some understanding and appreciation for the differences between hardware and software development. In addition to our

inability to identify user requirements, a second problem area is simply our unfamiliarity with the software development process. Typically, today's program manager was brought up though the ranks in one of the technical disciplines other than software engineering. Just ask any of them a hardware related question at a program review, and then stand by for a lengthy dissertation on every design detail. Ask that same manager a software related question and you will get the, "things are right on schedule" answer. Realizing that this whole software reliance environment is a relatively new field, it will still take some time before a true appreciation of the software design process is attained.

Watts S. Humphrey, of the Software Engineering Institute, outlines six common misconceptions about the software process. He lists:

- *We must start with firm requirements.* There is a widespread but fallacious view that requirements are the customers' job and that development should not start until they are explicitly defined.
- *If it passes test, it must be OK.* If the generally dismal record of software quality problems doesn't prove this false, nothing will.
- *Software quality can't be measured.* While not as well recognized, this too is false.
- *The problems are technical.* In spite of the many improved languages, tools, and environments, the problems of software cost, schedule, and quality remain.
- *We need better people.* Since software professionals make the errors, some people erroneously feel that they should be blamed for them.
- *Software management is different.* While this is a new and unique field, traditional management methods can and should be used. [Ref. 43: 25]

Mr. Humphrey further states that, "The remarkable thing about these misconceptions is that even though most software professionals recognize their

fallacy our contracts and our management systems are largely based on them." [Ref. 43:25]

As stated earlier, although recognized as the most effective and efficient way to develop software to meet new requirements, the prototyping approach basically conflicts with contracting practices. One of the things that Mr. Humphrey has done is to look at the mind-set of those in the procurement business. He looks at software contracting in terms of trust.

> How would buyers behave if they did not trust their vendors, doubted their competence, and believed in their own ? Under these conditions, they would probably do the following:
>
> 1. Start with a very precise statement of what was wanted.
> 2. Insist on rigid standards and detailed documentation of every step.
> 3. Require that each step be completed and approved before the next was initiated.
> 4. Demand a firm commitment at the outset. [Ref. 43: 411]

Humphrey is not implying that software developers are dishonest liars, cheaters or stealers, but rather, that trust is a relational attitude between program managers and contracting officers and the contractor. Like in any other relationship, trust is slow to evolve and quick to evaporate. Trust is developed over time as the two parties become familiar with each other. Humphrey, on the other hand, warns, "while trust is important, blind faith is not. There is, however, an enormous difference between an adversarial contract and one with a reasonable level of arm's-length protection" [Ref. 43: 413].

This basic principal of trust readily applies in the software contracting environment. It is unfortunate, however, that the entire software development environment is judged on those instances where success is not

optimum. In other words, successful software development programs are not news worthy. On the other hand, as highlighted in Chapter II, the "not so successful" development programs consume such a tremendous amount of resources that all programs tend to be approached with skepticism. This skepticism can have a positive benefit if it results in a management, inspection, quality, and test program that will audit developmental efforts. If done properly, the results will be an environment of trust, the win/win relationship that we strive to achieve.

C. SOFTWARE QUALITY

In Chapter I, quality was equated to the software's ability to meet the user's ultimate mission need. Although from the user's perspective this approach may be adequate, from a management perspective it is much too simplistic. The program manager and contracting officer must be able to track the program through all stages of development. Management's assessment of the development program cannot be based solely on the contractor's ability to satisfy user requirements and/or meet scheduled milestones. This approach would imply that the manager might not have a clue as to the real status of the software development program until the end product is delivered. It should be obvious that this "wait and see" approach is not very wise considering the consequences of a embedded software development program that suffers cost overruns and schedule delays. Software quality must therefore be defined in terms other than customer satisfaction. It is difficult, at best, to define and quantify measures of software quality.

Measures of quality can be grouped into five general classes: Development, Product, Acceptance, Usage, and Repair. Table VII [Ref. 43: 339]

TABLE VII. Classes of Quality Measures

	Objective	Timely	Available	Representative	Controllable
Development					
Defects	yes	yes	yes	moderate	yes
Change activity	yes	yes	yes	poor	no
Product					
Error seeding	moderate	yes	difficult	doubtful	moderate
Software structure	depends	yes	moderate	doubtful	yes
Controlled tests	moderate	yes	difficult	good	yes
Acceptance					
Problems	no	late	yes	good	moderate
Install effort	moderate	late	difficult	good	yes
Usage					
Problems	no	late	yes	good	moderate
Operating effort	moderate	late	difficult	good	yes
Surveys	no	late	difficult	very good	no
Availability	yes	late	moderate	very good	yes
Repair					
Defects	yes	late	yes	moderate	yes
Repair effort	moderate	late	moderate	moderate	yes

Source: Humphrey, Watts S. *Managing the Software Process*

depicts a matrix that characterizes of each of these categories with respect to objectivity, timeliness, availability, the degree to which the buyer believes it is good (representation), and controllability. The program manager must choose those criteria which will best serve the program's needs. He must then translate those criteria into contractual requirements. Those contractual requirements must, in turn, be in such a format and language as to allow their effective use. The program manager will undoubtedly have access to technical assistance, but simplicity and ease of use by nontechnical decision makers will remain the key to useful management measurements/tools. The intent, here, is not to provide a detailed explanation of each of the criteria. It is, however, to show that there are many criteria with varying degrees of appropriateness depending on the nature of the program for measuring

software quality. Each of the criteria can play a vital role in measuring program success. For example, one of the quality measuring criteria calls for the monitoring of defects during development. Commonly experienced error rates between 1 and 3 errors per 1000 lines of code (kloc) seem somewhat trivial, and in the past somewhat acceptable. Today's increased reliance on software to satisfy weapon systems' mission requirements, however, make these numbers unacceptable. It is estimated that the Strategic Defense Initiative (SDI) program will total over 25 million lines of code. Applying the 1-3/kloc error rate would mean between 25,000 and 75,000 errors in the operational and support program [Ref. 44: 93-94]. Management attention must be turned toward the process, not just the result. "To consistently achieve superior performance, management must establish challenging quality goals and strive to meet them" [Ref. 43: 358].

D. MANAGEMENT ASSESSMENTS

The first step in developing the relationship of trust, as previously discussed, is to establish a baseline with regard to a contractor's capabilities. Taking the Total Quality Management (TQM) approach, this means not only an assessment of the contractor's capabilities, but more importantly, an assessment of his organizational software development process. Quality is a direct reflection of the company's process. A concentrated effort/pressure should be applied on the contractor to improve his own process, thus, resulting in an improvement to quality. The Software Engineering Institute (SEI), Carnegie Mellon University, has issued two technical reports that can aid the program manager during this assessment.

1. A Method for Assessing the Software Engineering Capability of Contractors. (CMU/SEI-87-TR-23)

The purpose of this document is to, "facilitate objective and consistent assessments of the ability of potential DOD contractors to develop software in accordance with modern engineering methods" [Ref. 45: 1]. Requested by the U.S. Air Force, this effort was a result of the realization that the government must be able to effectively evaluate the capabilities of potential contractors. An extremely straight forward document, it provides a series of questions to evaluate the contractor's software development process. It is an outstanding tool for every program manager, contracting officer, and source selection participant. The intent of this program is to provide potential contractors with a questionnaire, based on this document. An assessment team is then sent to each of the contractors' facilities to address each of the questions in detail collecting data to make their final evaluation.

2. Conducting SEI-Assisted Software Process Assessments

This document describes the Software Engineering Institute's procedure for assessing a contractor's software process. It is not intended as a guide that can be used by the program manager, it does however, provide a description of the Software Engineering Institute's capability to review an organization's software process. The assessment will determine the software process that is being used, identify key areas for improvement, and provide the contractor with assistance for implementing and monitoring the changes. While this level of effort may not be appropriate for all programs, the program manager should consider an early assessment of the contractor's process. For many of the major development programs, this guide would be excellent. [Ref. 46]

E. DEPARTMENT OF DEFENSE REPORT ON MILITARY SOFTWARE

In late 1987, the Defense Science Board Task Force on Military Software forwarded its report to the Secretary of Defense. The Board reported on the various initiatives within the Department of Defense aimed at improving software development. They included the Ada programming standard, the Software Technology for Adaptable, Reliable Systems (STARS), the Strategic Computing Initiative, the Software Engineering Institute, and the Strategic Defense Initiative. the Executive Summary of this report stated:

> ...The big problems are not technical. In spite of the substantial technical development needed in requirements-setting, metrics and measures, tools, etc., the Task Force is convinced that today's major problems with military software development are not technical, but management problems...[Ref. 47: 1]

F. DEPARTMENT OF DEFENSE SOFTWARE MASTER PLAN

One step in the right direction is the publication of the Department of Defense's "Software Master Plan". Released in preliminary draft form during February, 1990, it is the Department of Defense's first effort to recognize and bring to light the unique problems associated with military software development. It begins with a description of the various activities that have a management role in software procurement. It begins, "In order to identify actions required to improve the software management process, an understanding of the current software management roles...within the DOD is essential" [Ref. 25: A-1]. The significant issue here is the realization that the "software management process" is unique entity, unlike hardware development, and the current situation requires top level management attention. Specific direction to improve the "process" is not provided, however, it can be easily deduced that this document will provide the basis for corrective action

within the DOD. It states, "...as evidenced by the information presented...the overall responsibility for software management is clearly fragmented across the DOD" [Ref. 25: A-1].

G. SUMMARY

This chapter has provided a broad overview of the many complex factors that a program manager must be aware of while managing a program involving software development. There must exist an awareness of the impacts of philosophical, environmental, and quality issues. Addressing these issues during the acquisition strategy formulation is a must.

The intent of this chapter is to show that even though the contracting world seems somewhat rigid and inflexible, there are many approaches that can be taken to improve software quality without having to change the contracting process. The significance is an emphasis on a change of the software development process, and not the contracting process.

V. DATA COLLECTION RESULTS

A. DATA COLLECTION SURVEY

Having completed an extensive review of the literature available, the next step was to see exactly how this "software development problem" is perceived in the field. In addition to the personal interviews that were conducted, a survey was forwarded to 150 management and contracting professionals from both the government and industry. Of the 65 responses received, 49 reported having had experience in programs that involved the development and acquisition of mission critical software. The average experience level of the respondents was 9.5 years, ranging from 30 years of experience to one year. A sample survey form is contained in Appendix A. Compiled survey results are contained in Appendix B.

There was a overwhelming consensus that a problem does indeed exist in the government's ability to procure custom designed software. The survey responses indicated that each of the three areas, technical, contracting, and program management equally shared the blame for the problems encountered. The following sections discuss each of the major survey areas.

B. PROBLEMS WITH SOFTWARE ACQUISITION

Respondents were asked to identify the percentage of software development programs that resulted in cost overruns, schedule delays and poor performance due to problems in one of three general areas: technical, contractual, managerial. Table VIII provides typical examples for each of the three

TABLE VIII. Three Categories of Problems in Software Development

AREA	EXAMPLE
TECHNICAL	Programming or interface difficulties
CONTRACTUAL	Contract interpretation difficulties, contract administration difficulties, or possibly the use of an inappropriate contract type.
MANAGERIAL	Improper acquisition strategies, bad business decisions, lack of attention, lack of planning

problem categories. For the purposes of this particular question, the respondents were also divided into three categories: technical personnel, contracting personnel, and management personnel.

1. Technical Problems

The survey showed a perception exists that 59 percent of software development programs suffered delays, cost overruns, and performance problems due to technical problems. Managers felt that 63.4 percent of the programs had technical problems, while the technicians themselves seemed to think it was more like 56 percent. Contracting personnel were right at the 59 percent level.

2. Contracting Problems

Overall, the respondents felt that 46.5 percent of software development programs suffered problems due to contracting issues. Technical and contracting personnel agreed that 48 percent of the programs were affected by contracting issues. Management, on the other hand, felt that only 42

percent of the programs suffered delays, cost overruns, and performance problems due to contracting issues.

3. Program Management Problems

Managers blew the whistle on themselves. They felt that 61.2 percent of the programs encountered problems due to "management related problems." The overall perception was 57.7 percent. Contracting personnel believed that only 50 percent of the programs suffered problems due to management issues, while the technical folks felt it was almost 59 percent.

4. Summary

Based on the data collected in this section of the survey, there is a perception that software development programs encounter delays, overruns, and performance problems almost 60 percent of the time due to technical issues, almost 58 percent of the time due to managerial, and less that 47 percent of the time due to contracting issues. A complete listing of the responses collected is contained in Appendix B.

C. TOP FIVE CONTRIBUTIONS TO DEVELOPMENT PROBLEMS

The next section of the survey asked respondents to identify the five, "...most common problems regarding contracting for and the management of a program involving software development". Table IX is a list of alternatives that were contained in the questionnaire. Answers were compiled using a simple weighting system to rank the responses. The ranking system assigned a weight of 10 points to the number one response, eight to number two, six to number three, four to number four, and two points to the number five response. The survey overwhelmingly showed that the respondents believed that the government's inability to identify clear user requirements was the

TABLE IX. Most Common Problem Areas in Software Development

Unclear user requirements	Unclear statement of work
Schedule too optimistic	Contract clauses too restrictive
Lack of gov't regulations	Too many gov't regulations
Inadequate gov't specifications	Too many gov't specifications
Lack of timely user feedback	Lack of design reviews
Wrong personnel at design reviews	Lack of hardware design freeze
Lack of software design freeze	Inadequate testing (gov't)
Failure to designate POC's	Inadequate testing (contractor)
Improper contract type	Difficult acquisition procedures
Lack of qualified gov't tech personnel	Inherent inability to measure reliability of software
Too many changes to requirements	Lack of contractor incentive
Inability to estimate cost	Lack of adequate documentation
Lack of understanding regarding software design and development	Lack of configuration mgmt
Mismanagement by industry	Mismanagement by gov't
	Others

number one problem in software development. The top five answers, along with their weightings are listed in Table X. A complete listing of all responses can be found in Appendix B.

TABLE X. Five Most Common Problems in Software Development

RANKING	PROBLEM	TOTAL POINTS
1.	UNCLEAR USER REQUIREMENTS	252
2.	TOO MANY CHANGES TO REQUIREMENTS	140
3.	LACK OF UNDERSTANDING REGARDING SOFTWARE DESIGN AND DEVELOPMENT	134
4.	SCHEDULE TOO OPTIMISTIC	130
5.	UNCLEAR STATEMENT OF WORK	92

1. Unclear User Requirements

As for what the respondents felt was at the heart of the number one problem, impressions varied greatly. "Users tend to not know what they want til they see it half done, then they are sure they want something different", said one respondent. On the other end of the spectrum, another felt that, "although the customer knows what he wants, he is unable to communicate his needs to the contractor..." Mr. Ted Bosworth, Vice President for C3 Systems at Comptek Research in Buffalo, NY, summed up the problem by saying, "The typically inaccurate initial translation between user and engineering communities, and the years between initial statement of need and completion of development, virtually assures that software 'forks' are delivered to provide users the ability to eat their 'soup'" [Ref. 48]. No matter what the underlying reason, a lack of knowledge or a lack of communication, it seems only logical that the government must develop a means to better identify end user requirements. Finally, one respondent stated, "Too many times the wrong people are asked what is required. If we could produce a rapid prototype to touch, feel and see maybe we would do a better job of it."

2. Too Many Changes to Requirements

Given that the government has problems identifying its requirements lends to the second problem area identified by survey respondents, "too many changes to requirements." On this subject one respondent said, "It is common to have several ECP's in work at several times all at different stages of definition, design, development or incorporation. This leads to multiple baselines which complicates... configuration management, schedules, estimating, and program management by both industry and government." Another expressed the difficulties with too many changes by making the analogy of trying to "hit a moving target."

3. Lack of Understanding Regarding Software Design and Development

Addressing this issue one respondent said:

> We (Govt) are not careful enough in our monitoring of processes that result in products/services. We traditionally have viewed (inspected) the finished product... In the case of software, this lack of attention to processes in design and development may fatally flaw the finished product and yet the flaw may go undetected for months/years.

There seems to be a consensus among respondents who chose this issue as one of their top five. One respondent felt that, "...almost everyone THINKS they understand software design and development and yet they do not communicate well with their counterparts in industry/government." Another respondent felt that, "issues such as 'Design for Supportability' and compatibility or interoperability get confused with ownership and cost/profit issues." It seems that this issue once again stresses the need for open lines of communication between the government and industry.

4. Schedule Too Optimistic

A survey respondent, a program manager in the private sector, who listed this as his number one problem said, "Typically the government acquisition process is so slow that it becomes mandatory to accelerate the development process in order to meet user needs. The complexity of the task is not re-scoped and this forces the contractor into attempting to achieve unrealistic schedules." Mr. Ted Bosworth stated:

> Every program I have been involved with that had significant software content had a schedule that assumed the user community knew exactly what they wanted; the Government development activity translated operational requirements into a technical specification completely, accurately, and without the possibility of misinterpretation by development engineers, and the engineers made no mistakes. Unfortunately for schedule performance, I have yet to see a set of humans that good. [Ref. 48]

5. Unclear Statement of Work

Many of these top five problems are interrelated. If the users fail to identify their needs accurately, then how can the program office write an adequate statement of work? One respondent stated, "Initial SOW's are what drive proposals — not what becomes the end item other than as an overall skeletal shell. The upgrading of the SOW is often not accomplished in a timely manner: hence there is an attempt to maintain proposal schedules which no longer reflect the realities." Another felt that, "many times the Statement of Work details are not clearly detailed in writing and it easily leads to misunderstandings on the part of the Contractor." On the other hand, one respondent felt that, "an unclear statement of work isn't necessarily all bad — especially with Full Scale Development programs... more emphasis should be placed on what the contractor intends to do and how they priced it." Still another respondent said, "...SOW's often have both too much detail and not enough. Too much detail limits the creativity of the contractor, and not enough results in a program that doesn't meet the need." In speaking about the contents of a statement of work, one respondent said, "Too many technical requirements are levied in lieu of user requirements, let the user define the requirements not the engineers interpreting the user's requirements."

D. CONTRACTING FOR SOFTWARE

1. Selection of a Contract Type

As discussed earlier in Chapter III, the selection of a contract type is a discretionary decision of the contracting officer and program manager, with some room for negotiation with the contractor. It is a decision that will hopefully result in a win-win situation between the government and the

contractor. Although many factors are involved in the decision process, the major factor is usually a question of risk assumption. Typically development programs, considered somewhat of a higher risk, would use some type of cost reimbursement arrangement. The opposite holds true in a more stable environment such as a follow-on or production effort where a fixed price arrangement is typically used.

One of the early premises of this thesis was that contract type might play a significant role in the eyes of the personnel involved in the procurement of custom designed software. This was shown to be untrue. Only three of the 49 respondents considered "improper contract type" as one of their top five concerns in software development programs. One respondent summed up this issue by stating:

> The contract type is not the key to success. A good contract is worth crap if the admin is poor. A bad contract can be a success if administered well. The problem is not the contract type it's how to make it work to meet your need. (Anonymous)

2. Advantages and Disadvantages of Contract Types

The two general categories of contract types that are available for software development are cost reimbursement and fixed price. Each has its advantages and disadvantages along with their appropriate uses.

It may be very appropriate to use a firm fixed-price contract for easily and accurately estimated software efforts. The biggest advantage of a firm fixed-price contract is there is very little administrative overhead. The contractor assumes the technical, schedule and cost risk. Once the contract is awarded the only concern becomes the deliverable product. On the other hand, a firm fixed-price contract may be the most inappropriate contract to use for software development, such as a new start weapons program. This type of pricing arrangement offers almost no flexibility to either the

contractor or the government. In a development environment, this means that the product is built to the specifications in the contract, period. If, as discussed above, the users have problems communicating their needs, the program office has problems interpreting their needs, and the engineers have problems satisfying those needs, the government and the contractor are locked into an agreement that isn't worth the paper its written on. The contractor is not incentivized to seek new, innovative, and possibly more efficient ways to satisfy software requirements. Likewise, the contractor has little motivation to take the extra time required to document the software other than the minimal requirements under the FFP agreement. This may lead to tremendous maintenance troubles during the systems life cycle. Contractors on a tight schedule and/or budget are less likely to restrict the level of effort put in to systems development. Most corporations would be very leery to sign a firm fixed price contract for those development programs that contain, in a prudent businessman's judgment, too many uncertainties.

The other option available for a software development effort is a cost reimbursement pricing arrangement. Under this arrangement, the contractor is relieved of the cost risk associated with a development program. The contractor has some flexibility as to how to satisfy the user's needs, and is not locked in to the single approach selected at the onset of the program. The major disadvantage lies in the fact that the contractor is entitled to all allowable and allocable expenses incurred. This means that the program office may have underestimated the task in terms of cost and time, resulting in a need for additional funding. Cost reimbursement contracts, in comparison to fixed price arrangements, require a great deal of administrative overhead.

3. Recommendations from the Field

Survey respondents were asked to identify the contract types that they had worked with during a software development effort. Responses covered the entire spectrum of contract types, bar none, outlined in the FAR. Survey respondents were then asked to identify the contract type that, "offers the most benefit to both the government and the contractor during a software development effort." A list of the recommended contract types is shown in Table XI. Eighteen of the respondents felt that they could not give an answer to the question. Most said that it was impossible to generalize, and the selection of contract type depended on the particular situation for each individual program.

TABLE XI. Recommended Contract Types for Software Development

CONTRACT TYPE	NO. OF RESPONSES
NO PARTICULAR RECOMMENDATION	18
COST-PLUS-AWARD-FEE	10
FIXED-PRICE-INCENTIVE	8
COST-PLUS-FIXED-FEE	6
COST-PLUS-INCENTIVE	3
COST TYPE (UNSPECIFIED)	2
FIRM-FIXED-PRICE	2

4. Complicating the Issue

It is very unlikely that a contract for the development of embedded mission critical software will be segregated from the hardware development effort. The selection of contract type now becomes a question of which contract will best suit the needs of the entire program.

If an incentive type contract is selected, the government must now decide just what to incentivize. Remember what was said about hardware earlier, we can see and feel it. It is much easier to structure an incentive contract around those things that can be seen and felt. This leaves a very difficult task for the contracting officer and program manager who attempt to structure the incentive type contract around software, as well as, hardware development measures.

It is not impossible to use incentive contracts for software development. It does, however, require a great deal of administrative and management effort. In an interview with Mr. James Swizewski, a contracting officer from the Naval Regional Contracting Center in Philadelphia, the subject of using a CPAF contract was discussed. He stated that, in fact, NRCC had used a CPAF contract for software development. He did not feel, however, that the contract was executed efficiently [Ref. 49]. Even though the contractual vehicle was in place, the contractor was not incentivized appropriately. A major draw back to the use of a CPAF contract is that the contractor will concentrate his efforts on those measures in the contract that will result in his receiving the award fee. In other words, regardless of those areas that may require special attention based on events during the contract performance, attention will go to areas defined at contract award that result in the receipt of the award fee. Contractors have been known to devote so much attention to the award fee criteria that the rest of the program suffers. Another drawback is the tremendous requirement for administrative and managerial overhead on the part of the government. Mr. Bud Wasgatt, Division Head of the Navy's Combat Systems Test and Analysis Department, Port Hueneme, CA, felt that the use of a CPAF contract is viable, but that the award fee must be tied to some software quality indicator [Ref. 50]. There lies the problem. Again, we

are back at the point of trying to identify and define software performance indicators.

E. INDEPENDENT VERIFICATION AND VALIDATION

One of the tools available to the program office is the use of an Independent Verification and Validation (IV&V) organization to monitor the software development process. The IV&V personnel are the government's counterparts to the company's Software Quality Assurance (SQA) staff. Mr. Humphrey best states the concept of IV&V:

> ...IV&V can and should capitalize on the existence of SQA. If SQA is working effectively, IV&V need not duplicate its work, and if not, IV&V must not try to replace it. Their role is to highlight this shortcoming and get it fixed. [Ref. 43: 151]

Duties of the IV&V staff include: design and coding analysis, approval of tests and test procedures, and witnessing of tests and inspections. Mr. Jimmie Carlisle, Atlantic Research Corporation, serves as the head of the IV&V team for the Marine Corps at Litton Data Systems Division in Los Angeles. Litton has developed and is currently producing the AN/TYQ-23, Tactical Air Operations Module, for the Marine Corps and the U.S. Air Force. He feels that the money dedicated up front by the Marine Corps to place his IV&V team in-plant will result in tremendous payback during the software operational and maintenance life cycle. Mr. Carlisle also sees an expansion of the role of IV&V in the future to include some types of independent testing. [Ref. 51]

The Marine Corps' SQA representative at the Marine Corps Tactical Systems Support Activity (MCTSSA), located at Camp Pendleton, CA, wholeheartedly concurs. CWO-3 David Mrazik feels that it is absolutely essential to dedicate resources during the development phase, independent of the

developing activity, to ensure supportability and maintainability throughout the software's life cycle. MCTSSA will assume software maintenance responsibility for the AN/TYQ-23 and its some 3.7 million lines of code (estimated) in 1992. [Ref. 52]

F. PERFORMANCE WARRANTIES

Another way that the government can insure itself, at least against monetary impacts, is to negotiate a warranty as part of the development/ production contract. Most typically used in the case of hardware deliverables, warranties protect against patent defects after government acceptance. (Patent defects are defects which should have been found through normal inspection.) Adding a slight twist to this concept, program offices are now using warranties to protect against software defects. The AN/TYQ-23 is such a program. A performance warranty is included in the contract stating that:

> The TACM Computer Programs which reside in the AN/AYK-14 or AN/UYK-44 Computer and the software/firmware embedded in the various unit processors...shall be free of performance defects during the period specified under PERFORMANCE WARRANTY PERIOD. A performance defect is defined as a failure of the computer program or embedded processor software/firmware which prevents or precludes the successful performance of, or accomplishment of, a function or sub-function as specified in the Development Specification...and the Program Performance Specifications... [Ref. 53: H-30]

The Performance Warranty Period is later defined as a period of, "two years commencing upon the acceptance of the First Article" [Ref. 53: H-31].

While the contract also includes the usual special provisions regarding actions that result in the warranty being "null and void", and the administration of such a warranty is going to be a great management challenge; this type of agreement is definitely a step in the right direction. [Ref. 54]

G. LESSONS LEARNED

Here are a few quotes that have been extracted from respondents' answers to the final two survey questions dealing with "lessons learned" and "other concerns":

- Educate both the customer and the contractor, during post-award, if not before, as to the real need for cooperation (and) understanding of the requirements
- Count on failure if your contracting officer is timid. If the system is an infantry system — make the PM infantry. If the system is for armor — make the PM armor... Keep the computer wizards out of the way. The systems they know work at the speed of light, but they don't.
- Don't sleep through the first six months assuming you can't get in much trouble so fast.
- Get it defined up front, what is wanted, when its wanted and how much.
- Close coordination between technical and contracting personnel during the development phase to assure only necessary changes are made.
- 1. Analyze the contract specification and develop a requirements document which identifies what you will implement for each require ment in the contract.

  2. Review your "Requirements Document" with the government so that early in the contract they will know how you have interpreted their requirements and how you intend to satisfy each one.

  3. Identify and record all disagreements and resolve them immediately.

  4. Then lay out a "doable" schedule and go to work.

  Do *not* wait until a CDR to tell the customer what he will receive as a product!
- Never assume that just because the government does not have technically qualified personnel that the contractor's personnel are more qualified than we are.
- Implement frequent and comprehensive in process reviews. Communications is the key.
- Management is illiterate when it comes to software. They will not admit it, therefore, they ignore it.

  1. Too much time spent studying
  2. Too little time spent applying
  3. Too much, "No, you can't do that"
  4. Too little, "Tell me more"

- Realistic time and cost budgets at the onset are essential for a successful software development program
- Managing software development is a fine balance between strict configuration management and control, and the freedom to let the designers be innovative and be personally involved in their work.
- Software is voodoo to most people and it's easy to get suckered (or bulled) into thinking you understand all the requirements of the technical community.
- Contract on a level of effort basis until all parties agree with the baseline requirements.
- An ill defined project cannot be monitored.
- It is very, very difficult to accurately estimate the cost and schedule for a software project that is unprecedented. Therefore the need to prototype difficult areas *before* awarding FSD contract.
- A customer who understands his requirement- in detail- and will assist in working the problem is irreplaceable.
- Risk management is the key to successful development programs
- Bright, experienced people that communicate effectively on both the Government and industry teams equals success. Drones on either side, or lack of communications, ensures failure.
- Identify development problems early. Force contractor management that meets intermediate objectives and milestones. Documentation is the most expensive segment of the development.
- Well defined functional requirements are imperative. On-going design review is critical. Good lines of communication between users, managers, and contractors is critical.
- Software is not something to be afraid of and it certainly should not be ignored. Many times software is ignored or played down due to sheer ignorance.

H. SUMMARY

Survey respondents felt that nearly 60% of all software development programs resulted in cost overruns, schedule delays and poor performance due to management and/or technical issues. They felt that almost 47% of the development programs suffered contractual problem. No matter where the blame lies, it is clear that both government and industry personnel have the perception that a problem exists in the government's ability to procure

custom designed software for its weapon systems. There is definitely a perceived, if not real, problem in the government's ability to procure customed designed software. Although there was tremendous diversity in the answers to questions asked during the data collection phase of this research, one underlying fact consistently came through — the need for open lines of communication.

VI. CONCLUSIONS AND RECOMMENDATIONS

" IF YOU ALWAYS DO
WHAT YOU ALWAYS DID
YOU WILL ALWAYS GET
WHAT YOU ALWAYS GOT "
CROMWELL (CIRCA 1650)
"AND WE DO, WE DO!"
SURVEY RESPONDENT (1990)

A. CONCLUSIONS

The research conducted has indicated that a serious problem exists within the Department of the Navy, and within the Department of Defense in general, concerning the procurement of customed designed software to be used within its weapon systems. The problem was first brought to the Government's attention in 1979 by the General Accounting Office in its report, "Contracting for Computer Software Development—Serious Problems Require Management Attention to Avoid Wasting Additional Millions." In 1987, the Defense Science Board Task Force on Military Software reported much the same situation. Their report to the Secretary of Defense began:

> Many previous studies have provided an abundance of valid conclusions and detailed recommendations. Most remain unimplemented. If the military software problem is real, it is not perceived as urgent. We do not attempt to prove that it is; we do recommend how to attack it if one wants to. [Ref. 47]

Ten years after the original GAO report, the same issues regarding the procurement of software were brought to the attention of Congress by their own Committee on Science, Space and Technology of the House of Representatives.

As discussed earlier in Chapter II, the major Department of Defense directives regarding weapon system acquisition do not recognize the critical importance of software procurement. This importance, and even more so reliance, on embedded software deserves special recognition in directives such as DOD Directive 5000.1, Major and Non-major Defense Acquisition Programs, and DOD Instruction 5000.2, Defense Acquisition Program Procedures. Embedded software, by the sheer volume of assets consumed in its design, development and maintenance, warrants special attention from a resource management stand-point. That management attention will occur only when the major DOD weapon system acquisition directives specifically address the unique requirements associated with software development. The Department of Defense Software Master Plan is a giant step in the right direction. More recognition must follow.

This research has indicated that this problem is recognized not only by external audit agencies such as the General Accounting Office, and Congressional committees, but also by the staff agencies within the Department of Defense itself. More importantly, the problems with embedded software are recognized by both the government and industry personnel who are charged with its development, procurement, and maintenance. Three-fourths of the engineers, contracting personnel, and managers who responded to this research's survey, felt that at least one half of the software development programs resulted in cost overruns, schedule delay and poor performance due to program management issues. They all recognized that software development programs typically have, in addition to management problems, technical and/or contractual problems. The top five most common problems identified by survey respondents, (unclear user requirements, too many changes to

requirements, lack of understanding regarding software design and development, schedule too optimistic, and unclear statement of work), all point directly at management issues vice problems in the technical or contracting areas.

How can a program office expect to ultimately satisfy the end user's mission need when that critical gap between the user and the software engineer is not bridged? This research has shown that, more than any other problem, the ability to accurately and correctly identify the user's requirements is the number one problem with software development today. There are, however, current development practices that can aid in building that bridge, specifically, the use of rapid prototyping.

Unfortunately, it is not an easy problem to correct. The heart of the problem does not lie in the complex system used to procure software. In other words, the answer to embedded software development is not in the contracting mechanism itself. The problem is fostered by the attitude and management decisions of the personnel within that procurement system. The answer lies in changing the "attitude", not in changing the basic procurement system. Our program managers and contracting officers must learn to make smart, well-informed decisions regarding software development.

C. RECOMMENDATIONS

The following recommendations are a result of this research effort:

1. Use of Rapid Prototyping

All significant software development programs should be contractually structured to require the iterative process known as "rapid prototyping." This process has become an accepted, and in most cases, the preferred

method of development to ensure the ultimate satisfaction of the user's needs. This may require a formal policy change.

Four of the top five areas of concern, identified by the survey respondents, would easily have less of an impact on software development programs if the use of "rapid prototyping" became the standard for development. Defining the user's requirements would become an iterative process, thus easing the pressure on the program manager and contracting officer to interpret these requirements into contractual language at the onset of the program. User involvement throughout the phases of development would be assured. This would also ease the pressure of detailing a statement of work in excruciating detail that contracting officers, managers, and engineers would be forced to live with throughout the program. Shorter periods of performance, through the use of intermediate milestones and user reviews, would make schedule setting more realistic. Finally, the impact of "too many changes to requirements" would be lessened because the intent of the prototyping effort would be to incrementally massage user requirements, taking the appropriate action to resolve any misinterpretations. The use of level of effort type contracts should be explored as a possible means to implement this prototype development methodology.

2. Use On-site IV&V on all Significant Software Development Efforts.

The use of independent technical and management teams to monitor the contractor's processes and performance is not an inexpensive undertaking. The benefits from budgeting resources at the front end of a program to do such a task, however, is without a doubt a worthwhile investment. Two requirements hold constant for all Independent Verification and Validation

(IV&V) efforts. First, the program office must be willing to dedicate the required assets to the IV&V team, basically in terms of appropriate staffing for the assigned task. It is foolish to halfheartedly provide a quasi-IV&V team. In the end, the program office may potentially find itself in twice as much trouble. Secondly, the IV&V team must be competent to perform the task. Probably the most obvious requirement, and at the same time, the hardest to satisfy. There are no magical answers or formulas to assess the competency of an IV&V team. This is made even more difficult because like many service contracts there are no tangible deliverable items. A third requirement which is almost a constant is the need for the IV&V effort to be collocated with the contractor. It is absolutely worth the time and money to have an on-site validation and verification team that reports directly to the program office. It is important that the program manager not get lulled into a false sense of security by using an IV&V team. Remember, they are technical people validating the technical approaches used by the contractor. Someone must still ensure that the software will satisfy the user's requirements.

3. Negotiate Software Performance Warranties in Development Contracts

Negotiating a performance warranty for embedded computer resources may be somewhat difficult, at best. There are definitely cost versus benefit trade-offs. The deliverable/operational software is only warranted against defects as delineated in the contract. As discussed earlier, a standard measurement of software quality is difficult to define. The contract may therefore call for the contractor produced Product Performance Specification (PPS) to be the basis for the warranty. In that case, the government is under the gun to ensure that the PPS, in fact, satisfies the users requirements.

Given the difficulties described above, it may still be in the best interest of the government to negotiate/require warranties for embedded software. The more common they become, the more contractors will be incentivized to specifically deliver products without defects, and to generally improve the overall quality of their software development process.

4. Education of Program Managers and Contracting Officers

This "attitude" regarding software development can only be changed through an educational process. In line with the initiatives to mandate educational requirements for contracting officers and program managers, a look at courses in software development management would be appropriate. The answer is not to make computer programmers contracting officers and program managers. It goes without saying, however, to make smart decisions management personnel must be somewhat literate with regard to software development techniques, processes, and the alternatives available. One way to aid in that literacy attainment is to establish specific, easily accessible, educational courses for those personnel that will manage and contract for software development.

D. ANSWERS TO RESEARCH QUESTIONS

1. What currently allowable contractual mechanisms can be used to improve the Department of the Navy's success in the procurement of custom developed software ?

As discussed in the conclusions and recommendations to this research effort, the current procurement problems regarding customed designed and developed software can all be addressed using management action within the current contracting system. The use of prototyping would require a re-thinking of the way we normally do business, but it is not only a

viable option, it is becoming the most valid approach. Acquisition strategies should call for the use of an on-site IV&V team from either dedicated in-house resources or procured from the private sector though a service type contract. The prime contractor, in this case must be made aware of the IV&V's relationship to the program manager and contracting officer, and the extent of their authority. The last contractual mechanism that should be seriously considered is the use of a performance warranty to both protect the government and incentivize the contractor.

While use of contracting mechanisms was the basic premise of this research effort, it became obvious that the problems encountered in the procurement of embedded computer resources was not a contracting issue.

2. How successful is the Department of Defense/Department of the Navy in the procurement of custom software?

Even based on the broad definition of program success given in Chapter I, this question remains difficult to answer. It is relatively easy to find program information for those programs that have suffered setbacks. The issue gains attention because of the nature of software development. It is manpower intensive, extremely costly, and delivers no tangible product. When setbacks are in terms of billions of dollars in overruns, years of slippage in schedules, and deliver of software that is not safe for use, the crisis seems phenomenal. Unfortunately, those development programs that are delivered on time, on budget, and that work, or suffer only minimal problems are not deemed news worthy in these times of public mistrust of government procurement. Successful programs do not get adequate recognition regardless of the product being developed.

3. What directives regulate and/or guide Program Managers and Contracting Officers in the procurement of custom software ?

The Brooks Bill (PL89-306) of 1965, and the Warner Amendment to the Fiscal Year 1982 Appropriations Act (PL 97-86) , are the two specific laws that pertain to software procurement. They mandate a clear set of directions regarding the procurement of computer hardware and software for administrative type functions and military missions, respectively. The Federal Acquisition Regulation (FAR) do not specifically recognize the procurement of software as being different from the procurement of any other good or service. The DOD FAR Supplement only provides guidance as far as recapitulating what type software procurement is allowable directly by the DOD under the Warner Amendment.

Within the Department of the Defense/Department of the Navy there are three major directives that provide guidance regarding the procurement and management of embedded computer resources. They are: DOD Directive 5000.29, Management of Computer Resources in Major Defense Systems; SECNAVINST 5200.32, Management of ECR in Department of the Navy Systems; and, MCO 5200.23A, Management of Mission-Critical Computer Resources (MCCR) in the Marine Corps. Numerous other orders and directives pertaining to specific areas of a system acquisition such as test and evaluation, configuration management, and standardization are also applicable to embedded computer resource development and procurement.

There are two military standards that provide the majority of the specific direction to the program manager and contracting officer. The first, DOD-STD-2167A, Defense System Software Development, delineates the requirements for mission critical software development. The second,

DOD-STD-2168, Defense System Software Quality Program, establishes the requirements for a contractor's software quality assurance program. These two standards are designed to be used in conjunction with each other.

4. What contract types are typically used for programs involving the development of custom software ?

There is no one contract type can be identified as "typically used" for software development. More importantly, there is no one contract type that can be identified as being the best for software development. The full spectrum of contract types, from Firm-Fixed-Price to Cost-Plus-Award-Fee contracts, were reported as having been used by the survey respondents. The research showed that every type contract has the potential for being used, and that the selection is totally dependent upon the situation.

5. Are there any contractual mechanisms that can be used to aid in the development of custom software ?

The provisions of DOD Standards 2167A and 2168 when called out in a contract are probably the best contractual direction that we can provide a contractor during a software development program. The use of IV&V teams and performance warranties will help to ensure the contractor is number one doing his job, and number two, willing to back the quality of his work. Most of the other influences pertaining to software development contract are managerial in nature. In other words, the government needs program managers and contracting officers who can make good business decisions while satisfying the user's mission requirements.

E. RECOMMENDATIONS FOR FURTHER STUDY

1. Formal Software Development Education

Examine the existing management schools within the Department of Defense to evaluate the content and the adequacy of courses offered dealing with the management and procurement of custom developed software. This research effort indicated that the root of the software development problem is not in the procurement system itself, but in the personnel within the system. As the dependence on software continues to grow, the ability of government procurement personnel to understand the nature of design and development of mission critical software will become increasingly more critical.

2. Use of "Rapid Prototyping" in Software Development

Examine various means to implement "rapid prototyping" as a standard development process for mission critical computer resources. This would involve an investigation of various contracting strategies that would support the use of software prototyping in development programs for weapon systems. As discussed above, the use of prototyping would go a long way in helping to resolve four of the five top software development issues identified in this research. In conjunction with the examination of contracting techniques, a review of existing policies dealing with the use of prototyping as an acquisition strategy should be undertaken. It may be useful to examine any programs that have used prototyping techniques for embedded software development, and develop a model program for software procurement based on this approach.

3. Use of Level of Effort Contracts in Software Development

Examine the use of level of effort type contracts for software development. In line with the recommendation above, one way of emulating a

prototype approach is to use progressive level of effort contracts, and/or competitive level of effort contracts. The Fleet Combat Direction System Support Activity at Dam Neck, VA, is doing just that. This research effort would include an analysis of their procedures for procuring custom software in the development phase, the production phase, and life cycle maintenance phase for a weapon system. [Ref. 55]

4. Software Performance Warranties

Examine the value of software performance warranties in weapons systems development and procurement contracts. Although the AN/TYQ-23, Tactical Air Operations Module, production contract has provisions that call out a two year software performance warranty, the system has yet to be fielded. The warranty is based on the system's compliance with the software Product Performance Specification. Looking back to Figure 4.3, one can see the relative cost of software changes that would be required to correct deficiencies. Therefore, one can easily deduce that the implementation of the warranty clause will be difficult, at best. Questions such as: in scope/out-of-scope?; system operated in accordance with operating manuals?; proper documentation of defects, and ability to recreate the defect?; will be negotiated well beyond the warranty period. This recommended research would analyze the effectiveness of those systems acquisition contracts that contain software performance warranties. Another approach would be to produce a model or a checklist for use as a guide when negotiating software warranties.

5. Model for a Weapon System Incentive Contract

Develop a model incentive type contract that can be tailored and used for weapon system procurement involving both hardware and software development. During the acquisition strategy formulation stages, it would be

extremely helpful to have a straw-man or model incentive contract to aid in the structuring of the contractual mechanism that would focus both hardware and software criteria. The ability and practicality of segregating software from hardware criteria would need to be investigated. The practicality and economics of using, for example, an award fee contract with incentives for software, as well as, hardware accomplishments would require analysis. This type approach would accomplish two valuable objectives. First, the contractor would be monetarily incentivized to place more attention on those software criteria deemed critical by the program office. Second, the program office would be forced (by the contractor seeking his monetary award) to pay closer attention to software development issues. This effort could also investigate the establishment of a model for the award criteria itself. In other words, it would research current software measures of effectiveness that could be appropriately tied to an incentive type contract.

APPENDIX A

SURVEY QUESTIONNAIRE

CONTRACTING FOR CUSTOM DEVELOPED SOFTWARE WITHIN DOD

1. What is your most recent job experience?

   A. Government _____ Industry _____

   B. Program Manager _____ Program Engineer _____
   Contracting Officer (PCO) _____ DCAS QA Rep _____
   Contract Administrator _____ Tech Rep _____
   Project Officer _____ Other _____
   (specify)

2. Have you been involved in a program that required the development of custom software (i.e., major weapon system) for the Department of Defense?

   Yes ____ No ____ **If YES please CONTINUE—If NO, STOP and kindly return the survey.**

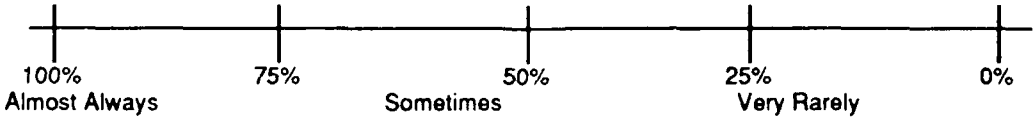
3. How many years experience do you have with acquisition programs requiring custom software?_____

4. Where would you say that the majority of your expertise lies?

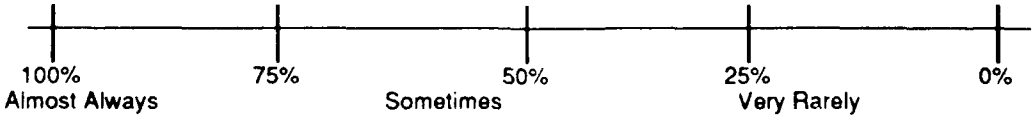
   Program Manager _____ Program Engineer _____
   Contracting Officer (PCO) _____ DCAS QA Rep _____
   Contract Administrator _____ Tech Rep _____
   Project Officer _____ Other _____
   (specify)

The following three questions deal with problems regarding computer software within a complex weapon system. For each question, place an 'x' along the corresponding scale.

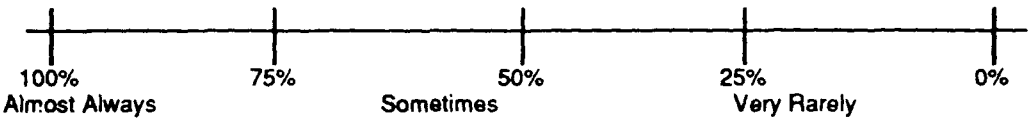
5. Do technical developmental problems (i.e., programming/interface difficulties) lead to cost overruns, schedule delays and poor performance?



6. Do contracting related problems lead to cost overruns, schedule delays and poor performance?



7. Do program management related problems lead to cost overruns, schedule delays and poor performance?



8. Please rank the FIVE (1 thru 5) most common problems regarding contracting for and the management of a program involving software development (with 1 being the most common problem)

Unclear user requirements	____	Unclear Statement of Work	____
Schedule too optimistic	____	Contract clauses too restrictive	____
Lack of gov't regulations	____	Too many gov't regulations	____
Inadequate gov't specifications	____	Too many gov't specifications	____
Lack of timely user feedback	____	Lack of design reviews	____
Wrong personnel at design reviews	____	Lack of hardware design freeze	____
Lack of software design freeze	____	Inadequate testing (gov't)	____
Failure to designate POCs	____	Inadequate testing (contractor)	____
Improper contract type	____	Difficult acquisition procedures	____
Lack of qualified gov't tech personnel	____	Inherent inability to measure reliability of software	____
Too many changes to requirements	____	Lack of contractor incentive	____
Inability to estimate cost	____	Lack of adequate documentation	____
Lack of understanding regarding software design & development	____	Lack of configuration mgmt	____
Mismanagement by Industry	____	Mismanagement by gov't	____
Others _____	____	_____	____
_____	____	_____	____

IF ADDITIONAL SPACE IS REQUIRED FOR ANY OF THE FOLLOWING QUESTIONS, PLEASE ATTACH A SEPARATE SHEET OF PAPER. HAND WRITTEN CONTRIBUTIONS ARE FINE.

9. Expand your thoughts and experiences regarding the top three items that you chose in the preceding question.

#1 _____

_____

_____

_____

_____

_____

#2 _____

_____

_____

_____

_____

#3 _____

10. What do you feel is the most important ingredient to the success of a software development phase in a major weapon system procurement from a contracting aspect?

11. What do you feel is the most important ingredient to the success of a software development phase in a major weapon system procurement from a management aspect?

12. What type contracts have you worked with on software development programs?
Firm Fixed Price ____ Fixed Price Incentive ___ Cost Plus Fixed Fee ___ Cost Plus Award Fee ___
Other _____

13. Have you experienced any particular benefits or drawbacks associated with any of the contract types mentioned in question 12 above?

14. What contract type do you feel offers the most benefit to both the government and the contractor during a software development effort?

15. What is the most important lesson you have learned (from either a good or bad experience) in dealing with a contract that requires software development?

16. Do you have any other concerns regarding the procurement of custom developed software?

THANK YOU FOR YOUR HELP AND COOPERATION

APPENDIX B

SURVEY DATA

NO.	GOVT	INDUSTRY	POSITION	YRS	TECH	CONTRACT	MGMT
1		X	SQA SUPERVISOR	4	80	50	50
2	X		CONTRACT ADMIN	2	70	25	35
3		X	GOV'T CONTRACTS MGR	10	20	70	85
4	X		PROGRAM ENGINEER	3	80	60	60
5	X		CONTRACT ADMIN	5	50	50	50
6		X	CONTRACT ADMIN	8	100	75	25
7	X		PCO	8	45	15	55
8		X	PROGRAM MANAGER	30	80	65	50
9		X	TEST MGR	3	80	80	50
10	X		CONTRACT ADMIN	5	85	70	50
11		X	PROGRAM ENGINEER	8	50	25	65
12	X		PCO	4	75	70	50
13	X		TEST DIRECTOR	8	75	20	70
14	X		PROGRAM ENGINEER	22	20	20	20
15	X		PROGRAM MANAGER	22	90	65	90
16	X		DCASPRO ENGINEER	5	20	15	15
17	X		COMPUTER ENGINEER	15	70	85	55
18	X		PROGRAM ENGINEER	2	70	65	65
19	X		PROGRAM ENGINEER	7	80	80	35
20		X	TECHNICAL REP	10	100	65	90
21	X		DCAS (DCMR) ENGINEER	10	50	65	50
22	X		CONTRACT ADMIN	8	35	50	75
23	X		SQA	13	60	50	75
24		X	SQA	7	75	70	85
25	X		PROGRAM MANAGER	7	30	20	40
26	X		PROGRAM MANAGER	25	75	25	45
27		X	CONTRACT ADMIN	3	45	60	65
28	X		TECHNICAL REP	10	75	50	65
29		X	PROGRAM ENGINEER	13	75	75	75
30	X		PROGRAM ENGINEER	5	45	90	80
31	X		QA REP	10	40	20	45
32	X		PROGRAM ENGINEER	20	50	35	75
33		X	PROGRAM ENGINEER	23	15	25	10
34		X	PROGRAM MANAGER	12	80	50	75
35		X	V.P. C3 SYSTEMS	20	40	70	70
36	X		PROGRAM MANAGER	7	75	25	50
37	X		PROJECT OFFICER	3	75	50	100
38	X		SYS ANALYST	2	75	20	65
39	X		PROJECT OFFICER	1	50	25	25
40	X		SOFTWARE ENGINEER	4	65	70	60
41	X		PROJECT OFFICER	4	80	50	50
42	X		COTR	4	75	25	25
43	X		PROGRAM MANAGER	10	50	20	75
44	X		QA REP	17	20	0	90
45		X	PROGRAM MANAGER	8	25	40	50
46		X	PROGRAM MANAGER	10	30	25	60
47		X	CONTRACT OFFICER	14	50	20	40
48		X	COMPUTER ENGINEER	12	20	40	65
49	X		PROJECT OFFICER	5	80	45	80

TECHNICAL

NO.	GOVT	INDUSTRY	POSITION	TECH	CONTRACT	MGMT	CONTRACT
15	X		COMPUTER ENGINEER	70	85	55	
12		X	COMPUTER ENGINEER	20	40	65	CPAF
10	X		DCAS (DCMR) ENGINEER	50	65	50	CPFF
5	X		DCASPRO ENGINEER	20	15	15	CPAF
23		X	PROGRAM ENGINEER	15	25	10	
22	X		PROGRAM ENGINEER	20	20	20	
20	X		PROGRAM ENGINEER	50	35	75	CPFF
13		X	PROGRAM ENGINEER	75	75	75	
8		X	PROGRAM ENGINEER	50	25	65	
7	X		PROGRAM ENGINEER	80	80	35	CPAF
5	X		PROGRAM ENGINEER	45	90	80	
3	X		PROGRAM ENGINEER	80	60	60	
2	X		PROGRAM ENGINEER	70	65	65	FPI
17	X		QA REP	20	0	90	
10	X		QA REP	40	20	45	FPI
4	X		SOFTWARE ENGINEER	65	70	60	CPAF
13	X		SQA	60	50	75	FPI
7		X	SQA	75	70	85	
4	X		SQA SUPERVISOR	80	50	50	CPFF
2	X		SYS ANALYST	75	20	65	FFP
10		X	TECHNICAL REP	100	65	90	FPI
10	X		TECHNICAL REP	15	50	65	
10.1			**AVERAGES**	56.14	48.9	58.9	

CONTRACTING

NO.	GOVT	INDUSTRY	POSITION	TECH	CONTRACT	MGMT	CONTRACT
8		X	CONTRACT ADMIN	100	75	25	CPIF
8	X		CONTRACT ADMIN	35	50	75	
5	X		CONTRACT ADMIN	85	70	50	FPI
5	X		CONTRACT ADMIN	50	50	50	
3		X	CONTRACT ADMIN	45	60	65	CPAF
2	X		CONTRACT ADMIN	70	25	35	COST PLUS
14		X	CONTRACT OFFICER	50	20	40	
4	X		COTR	75	25	25	CPAF
10		X	GOVT CONTRACTS MGI	20	70	85	CPAF
8	X		PCO	45	15	55	COST TYPE
4	X		PCO	75	70	50	
6.45			**AVERAGES**	59.09	48.18	50.46	

MANAGEMENT

NO.	GOVT	INDUSTRY	POSITION	TECH	CONTRACT	MGMT	CONTRACT
15	X		COMPUTER ENGINEER	70	85	55	
30		X	PROGRAM MANAGER	80	65	50	CPFF
25	X		PROGRAM MANAGER	75	25	45	CPIF
22	X		PROGRAM MANAGER	90	65	90	
12		X	PROGRAM MANAGER	80	50	75	CPIF
10	X		PROGRAM MANAGER	50	20	75	
10		X	PROGRAM MANAGER	30	25	60	
8		X	PROGRAM MANAGER	25	40	50	CPAF
7	X		PROGRAM MANAGER	75	25	50	CPFF
7	X		PROGRAM MANAGER	30	20	40	FPI
5	X		PROJECT OFFICER	80	45	80	FPI
4	X		PROJECT OFFICER	80	50	50	CPAF
3	X		PROJECT OFFICER	75	50	100	FFP
1	X		PROJECT OFFICER	50	25	25	
8	X		TEST DIRECTOR	75	20	70	FPI
3		X	TEST MGR	80	80	50	CPFF
20				40	70	70	
10.9			**AVERAGES**	63.44	42.19	61.25	

PERCENTAGE OF SOFTWARE DEVELOPMENT PROGRAMS THAT SUFFER PROBLEMS AS A RESULT OF TECHNICAL, CONTRACT, OR MANAGEMENT ISSUES

	PERCENTAGE OF PROGRAMS		
RESPONDENT	TECHNICAL	CONTRACT	MANAGEMENT
DCAS SQA	80	50	50
COMMANDER DCASPRO	70	30	40
PROCUREMENT MANAGER	20	70	85
PROGRAM ENGINEER	80	60	60
CONTRACT ADMINISTRATOR	50	50	50
CONTRACT ADMINISTRATOR	100	75	25
CONTRACTING OFFICER	50	10	55
PROGRAM ENGINEER	80	65	50
TEST DIRECTOR/MANAGER	85	80	50
CONTRACT ADMINISTRATOR	85	70	50
SYSTEMS ENGINEER	50	25	70
COMMANDER DCASPRO	75	70	55
TEST DIRECTOR	75	20	70
NAVPRO ENGINEER	35	50	75
DCAS ENGINEER	50	65	50
PROGRAM MANAGER	30	20	45
PROGRAM MANAGER	75	25	40
COMPUTER ENGINEER	65	85	55
PROGRAM ENGINEER	70	65	65
PROGRAM ENGINEER	80	80	35
PROGRAM MANAGER	90	65	90
CONTRACT MANAGER	45	65	60
V.P. C3I SYSTEMS	40	75	75
TECH REP	75	50	70
PROGRAM ENGINEER	75	75	75
PROGRAM ENGINEER	45	85	80
QA REP	40	20	45
PROGRAM ENGINEER	20	20	20
PROGRAM ENGINEER	50	40	75
PROGRAM ENGINEER	15	25	5
PROGRAM ENGINEER	85	50	75
DCASPRO ENGINEER	15	15	15
PROGRAM MANAGER	50	35	75
PROJECT OFFICER	50	25	25
PROGRAM MANAGER	75	20	65
CONTRACT ADMINISTRATOR	75	50	50
PROGRAM MANAGER	75	25	50
PROJECT OFFICER	75	50	100
AVERAGE	61.83	49.88	57.80

RANKING	MOST COMMON PROBLEM IN SOFTWARE DEVELOPMENT	WT. RESPONSE
1	Unclear user requirements	252.0
2	Too many changes to requirements	140.0
3	Lack of understanding regarding software design	134.0
4	Schedule too optimistic	130.0
5	Unclear statement of work	92.0
6	Lack of qualified gov't tech personnel	88.0
7	Inadequate gov't specifications	76.0
8	Lack of software design freeze	66.0
9	Inability to estimate cost	54.0
10	Lack of adequate documentation	46.0
11	Mismanagement by the government	40.0
12	Difficult acquisition procedures	36.0
13	Inadequate testing (contractor)	28.0
14	Mismanagement by Industry	28.0
15	Lack of timely user feedback	26.0
16	Lack of configuration mgmt	26.0
17	Lack of contractor incentive	26.0
18	Lack of hardware design freeze	24.0
19	Wrong personnel at design reviews	24.0
20	Too many gov't regulations	24.0
21	Improper contract type	16.0
22	Inability to measure reliability of software	16.0
23	Too many gov't specifications	12.0
24	Contract clauses too restrictive	10.0
25	System Engineering failures	10.0
26	Inadequate source selection	10.0
27	Inadequate testing (gov't)	6.0
28	Lack of design reviews	4.0
29	Failure to designate POCs	2.0
30	Inability to measure software maintainability	2.0
31	Lack of gov't regulations	0.0

APPENDIX C

LIST OF PERSONNEL INTERVIEWED

1. Carlisle, Jimmie. Technical Manager, Defense Systems Division, Atlantic Research Corporation, Van Nuys, California, Interview, April 1990.

2. Bosworth, Ted. Vice President for C3 Systems, COMPUTEK Research Inc., Buffalo, New York, Interview , May 1990.

3. Firestone, David. Acquisition Engineer, AN/TYQ-23 Tactical Air Operations Module, Space and Naval Warfare Command, Washington, D.C., Interview, March 1990.

4. Mrazik, David R. CWO-3 USMC. Software Quality Assurance Officer, Marine Corps Tactical Systems Support Activity, Camp Pendleton, California, Interview, April 1990.

5. Primm, Ed. Fleet Combat Direction System Support Activity, Dam Neck, Virginia, Interview, May 1990.

6. Quattlebaum, John R. Lt Col USMC. Deputy Program Manager, AN/TYQ-23 Tactical Air Operations Module, Space and Naval Warfare Systems Command, Washington, D.C., Interviews, January and March 1990.

7. Swizewski, James. Contracting Officer, Naval Regional Contracting Center, Philadelphia, Pennsylvania, Interview, March 1990.

8. Thomas, Bruce J. Major USMC. Naval Electronic Systems Command Technical Representative, Litton Data Systems, Van Nuys, California, Interview, April 1990.

9. Wasgatt, Bud. Division Head, Combat Systems Test and Analysis Department, Naval Ship Weapon Systems Engineering Station, Port Hueneme, California, Inteview, April 1990.

LIST OF REFERENCES

1. Garvin, David A. "What Does 'Product Quality' Really Mean?" *Sloan Management Review*. Fall 1984.

2. Comptroller General of the United States. *Contracting for Computer Software Development — Serious Problems Require Management Attention to Avoid Wasting Additional Millions*. Report to the Congress. 9 November 1979.

3. Subcommittee on Investigations and Oversight, U.S. House of Representatives. *Bugs in the Program, Problems in Federal Government Computer Software Development and Regulation* . Washington D.C. 1989.

4. Kitfield, James. "Is Software DOD'S Achilles' Heel?" *Military Forum* July 1989.

5. Glaseman, Steven. *Comparative Studies in Software Acquisition*. Lexington: Lexington Books, 1982.

6. Burke, Cpt John D. USA. "Software Testing Management" *Program Manager* May-June 1988: 58-68.

7. Glass, Robert L. *Software Reliability Guidebook*. Englewood Cliffs: Prentice-Hall, 1979.

8. United States Navy. *Navy Program Manager's Guide,* 1987.

9. United States General Accounting Office. *NORAD's Communications System Segment Replacement Program Should Be Reassessed*. Report to the Subcommittee on Defense, Committee on Appropriations, House of Representatives. November 1988.

10. United States General Accounting Office. *Space Defense, Management and Technical Problems Delay Operations Center Acquisition*. Report to the Subcommittee on Defense, Committee on Appropriations, House of Representatives. April 1989.

11. United States General Accounting Office. *Battlefield Automation, Army Tactical Command and Control System's Cost and Schedule*. Briefing Report to the Subcommittee on Defense, Committee on Appropriations, House of Representatives. February 1990.

12. United States General Accounting Office. *Battlefield Automation, Army's Air Defense Command and Control System Status and Program Issues.*, Briefing Report to the Subcommittee on Defense, Committee on Appropriations, House of Representatives. Dec 1989.

13. Steigman, David. S. "Seawolf's Combat System Missing Deadline" *Navy Times* 9 April 1990: 25.

14. Geving, Donald L., *Management of MIFASS, a Marine Corps C2 System,* Master's Thesis, Naval Postgraduate School, Monterey, California, September 1987.

15. Waters, Robert. "Marines, Norden lose millions on failed computer" *Record-Journal*, Meriden, CT. August 9, 1987: p. A-1.

16. *Congress and the Nation Volume II 1965-1968.* Congressional Quarterly Service, Washington D.C. 1969.

17. United States. *United States Statues at Large 1965.* U.S. Government Printing Office, Washington D.C. 1966.

18. United States Congress. Fiscal Year 1982 DOD Appropriations Act. Public Law 97-86. 1981.

19. Department of Defense. *DOD Supplement to the Federal Acquisition Regulation,* 1988.

20. Department of Defense Directive 5000.1, *Major and Non-Major Defense Acquisition Programs,* 1 September 1987.

21. Department of Defense Directive 5000.2, *Defense Acquisition Procedures,* 1 September 1987.

22. Department of Defense Directive 5000.3, *Test and Evaluation,* 28 April 1989.

23. Department of Defense Directive 5000.29, *Management of Computer Resources in Major Defense Systems*, 28 December 1976.

24. Secretary of the Navy Instruction 5200.32, *Management of ECR in Department of the Navy Systems*, 11 June 1979.

25. Department of Defense. *Software Master Plan (Preliminary Draft).* 9 February 1990.

26. United States Marine Corps. *Systems Acquisition Management Manual* (MCO P5000.10), 1989.

27. United States Marine Corps MCO 5200.23A, *Management of Mission-Critical Computer Resources (MCCR) in the Marine Corps*, 30 December 1986.

28 Department of Defense. DOD-STD-1467, *Software Support Environment,* 18 January 1985.

29. Department of Defense Standard DOD-STD-1703 (NS), *Military Standard Software Product Standards,* 12 February 1987.

30. Department of Defense Standard DOD-STD-2167A, *Military Standard Defense System Software Development*, 29 February 1988.

31. Department of Defense Standard DOD-STD-2168, *Defense SystemSoftware Quality Program*, 29 April 1988.

32 Department of Defense Military Handbook, MIL-HDBK-286, *Software Quality Evaluation*, 31 December 1985.

33. Department of Defense Military Handbook, MIL-HDBK-287, *A Tailoring Guide for DOD-STD-2167A, Defense Systems Software Development*, 11 August 1989.

34. Department of Defense Military Handbook, MIL-HDBK-782, *Software Support Environment Acquisition*, 29 February 1989.

35. Department of the Navy. *RDT&E/Acquisition Management Guide*. 1989.

36. Defense Systems Management College. *Acquisition Strategy Guide*. 1984.

37. United States Government. *Federal Acquisition Regulation (FAR).* 1984.

38. Bergantz, LTC Joseph L. USA. "Alternative Contracting/Acquisition Strategies Within DOD". *Program Manager* September-October 1989: 32-36.

39. Francom, Gerald Lee., *A Proposal to Change the Federal Acquisition Regulation: Recognizing the Award Fee Incentive in Fixed-Price Contracts*, Master's Thesis, Naval Postgraduate School, Monterey, California, June 1989.

40. Garrett, Gregory, A. "Contracting with an Award Fee — It Works !" *Program Manager*. May-June 1989: 26-28.

41. Gordon, Harvey J. "The Role of the Contract in Systems Acquisition" *Concepts.* Winter 1980.

42 Pressman, Roger S. *Software Engineering.* McGraw-Hill Book Co., 1987.

43. Humphrey, Watts S. *Managing the Software Process,* Addison-Wesley Publishing Company, 1989.

44. Myers, Ware. "Software Pivotal to Strategic Defense" *Computer.* January 1989.

45. Software Engineering Institute Technical Report CMU/SEI-87-TR-23, Carnegie Mellon University, *A Method for Assessing the Software Engineering Capability of Contractors,* by Watts S. Humphrey and W.L. Sweet, September 1987.

46. Software Engineering Institute Technical Report CMU/SEI-89-TR-7, Carnegie Mellon University, *Conducting SEI-Assisted Software Process Assessments,* by Timothy G. Olson, Watts S. Humphrey, and David H. Kitson, February 1989.

47. Defense Science Board Task Force. "Report on Military Software" September 1987.

48. Telephone conversation between Mr. Ted Bosworth, Vice President for C3 Systems, Comptek Research Inc., Buffalo, NY., and the author, 29 May 1990.

49. Interview between Mr. James Swizewski, James, Contracting Officer, Naval Regional Contracting Center, Philadelphia PA., and the author, March 1990.

50. Interview between Mr. Bud Wasgatt, Division Head, Combat Systems Test and Analysis Department, Naval Ship Weapon Systems Engineering Station, Port Hueneme, and the author, 16 April 1990.

51. Interview between Mr. Jimmie Carlisle, Technical Manager, Defense Systems Division, Atlantic Research Corporation, and the author, 17 April 1990.

52. Interview between CWO-3 David R. Mrazik USMC, Software Quality Assurance Officer, Marine Corps Tactical Systems Support Activity, Camp Pendleton, CA. , and the author, 18 April 1990.

53. Naval Electronic Systems Command, AN/TYQ-23, Contract Number N00039-87-0330, dated 29 May 1987.

54. Telephone conversation between David Firestone, AN/TYQ-23 (V) Acquisition Engineer, and the author ,15 March 1990.

55. Telephone conversation between Mr. Ed Primm, Fleet Combat Direction System Support Activity, Dam Neck, VA, and the author, 18 May 90.

BIBLIOGRAPHY

Department of Defense. *Military Specification Software Quality Assurance Program Requirements*. MIL-S-52779A. 1 August 1979.

Fiegenbaum, A.V. *Total Quality Control*. McGraw Hill Book Co., 1983.

Hall, Patrick A.V. "Software Development Standards." *Software Engineering Journal* May 1989: 143-147.

King, David. *Current Practices in Software Development*.. Yourdon Press, 1984.

Kitchenham, Barabara A. and John G. Walker. "A Quantitative Approach to Monitoring Software Development." *Software Engineering Journal*, January 1989: 2-13.

Laski, Janusz. "Testing in the Program Development Cycle." *Software Engineering Journal,* March 1989: 95-105.

Marriott, Philip C. "Guidelines for Acquiring Quality Software from Software Vendors." *Total Systems Reliability Symposium: IEEE Computer Society.* December 1983.

Morrison, LCdr R.W. USNR. "Some Tips on Software Acquisition Management." *Program Manager*, September-October 1987: 42-46.

Pontius, Harry E. *Life Cycle Guidelines for Weapon System Software Management*. Defense Systems Management School. May 1976.

Rosenthal, Lynne S. "Planning and Implementing System Reliability" *Total Systems Reliability Symposium: IEEE Computer Society*. December 1983.

Schneidewind, Norman F. "Software Maintenance: Improvement Through Better Development Standards and Documentation." Naval Postgraduate School, Monterey. February 1982.

Schneidewind, Norman F. *Usability of Military Standards for the Maintenance of Embedded Computer Software*. Naval Postgraduate School, Monterey. June 1982.

Spatola, Michael, A. *Contracting for Weapons System Software*. Air Command and Staff College. 1985.

United States General Accounting Office. "ADP Budget, Department of the Navy's Information Technology Systems Obligations." *Report to the Congress.* May 1989.

United States General Accounting Office. "DOD Acquisition, Information on Joint Major Programs." *Report to the Congress.* July 1989.

United States General Accounting Office. "Programming Language, Status, Costs, and Issues Associated With Defense's Implementation of Ada." *Report to the Congress.* March 1989.

United States General Accounting Office. "Space Defense, Management and Technical Problems Delay Operations Center Acquisition." *Report to the Congress.* April 1989.

Wohlin, C. and Vrana, C. "A Quality Model to be Used During the Test Phase of the Software Lifecycle." Institute of Electrical Engineers Sixth International Conference. April 1986.